QUANTIFYING THE SOFTWARE MAINTENANCE TASK:

AN EMPIRICAL STUDY OF

COMPLEXITY METRICS

ACROSS VERSIONS

By

CHARI K. NANDAKUMAR

Bachelor of Engineering

Madurai Kamaraj University

Madurai, India

1985

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May 1989

QUANTIFYING THE SOFTWARE MAINTENANCE TASK:

AN EMPIRICAL STUDY OF

COMPLEXITY METRICS

ACROSS VERSIONS

Thesis Approved:

M. Samadzadeh-H

Thesis Adviser

S. E. Hedrick

J P Chandler

Norman N. Durham

Dean of the Graduate College

PREFACE

This study concerns the discussion of popular software metrics, models and their applications. The thesis presents an empirical investigation of selected software metrics and also an empirical validation for a new metric, the residual complexity metric, based on industrial data.

I owe much appreciation and gratitude to my major adviser Dr. Mansur H. Samadzadeh for his continuous guidance, motivation, and valuable instructions throughout this study.

I extend my appreciation to Drs. George E. Hederick and John P. Chandler for their suggestions and advice while serving on my committee.

My sincere thanks are extended to TMS Inc., Stillwater, especially to Mr. Art Crotzer for granting me the privilege of using their software as data and providing me useful resources for this study. Finally I extend my appreciation to Mr. Farrell McGoohan and Mr. Mike Christian for patiently filling out the survey forms.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

The field of software metrics and software models has gained much popularity over the past few years. The software development models that have been suggested [8,10,36] may be used in a number of ways by software managers. The models can be used to manage software development, testing and maintenance. There are also many potential uses of software metrics. Software metrics can be used to measure the psychological complexity of software maintenance tasks [12]; to evaluate the software development process [1], to predict faults in software [4,12]; to analyze programs [37]; and to give broad guidelines to programmers. The metrics also can be used to model development time, effort, development cost and maintenance cost [19,20].

1.1 Software Metrics

Software metrics are used mainly to characterize the conceptual complexity of software. Complexity (computational or conceptual) can be defined as a [13]

> characteristic of the software interface which influences the [machine or human] resources the system will expend or commit while interacting with the software.

There are many ways in which software metrics can be classified. The major classification scheme suggested by Conte et al. [8] is to divide the metrics into process metrics and product metrics. Process metrics are applied to the development process while the product metrics capture some measurable characteristics of the software. The product metrics are suited to automation. Some of the product metrics, namely size metrics, token counts, McCabe's metrics and residual complexity metrics are explained below. Some of the other metrics that have received attention are the amount of data [8], variable spans [14], fan-in of program modules [24], minimum number of paths in a program and reachability of any node [37], nesting-levels [43] and interconnectivity [26].

(Notational Convention: In the following text, the logarithm is assumed to be base 2)

1.1.1 Size Metrics

The "size" of a program is an important measure of the program and constitutes the most important factor for many models of software development. There are many possibilities for measuring the size of a program. For example one size metric can be the number of lines of code. Other size metrics may be a count of the tokens, or a count of function points [1].

The most widely used size metric is the count of lines of code (LOC). It will be represented by S_s in this report. S_s can be formally defined as [8],

> a line of code is any line of program text that is not a comment or blank line, regardless of the number of statements or fragments of statements on the line.

1.1.2 Token Counts

M. H. Halstead viewed a program as a sequence of tokens. The tokens either could be operands representing data or they could be operators manipulating the operands. He based his work [18,19,20] on various measures of operators and operands. Halstead's four basic counts are

n_1 : Number of unique operators

n_2 : Number of unique operands

N_1 : Total occurrences of operators

N_2 : Total occurrences of operands

There is no general agreement regarding exactly which tokens in a given language are operators and which are operands.

Halstead also defined various metrics based on these four basic counts, some of which are listed below.

The vocabulary n is defined as

$$n = n_1 + n_2$$

The program length N is given by

$$N = N_1 + N_2$$

The estimated program length is defined by

$$N_{est} = n_1 * \log(n_1) + n_2 * \log(n_2)$$

The measure program volume V is given as

$$V = N * \log(n) \quad \text{bits}$$

The potential volume of an algorithm V^* is the volume of that algorithm for its minimal implementation given by

$$V^* = (2 + n_2^*) * \log(2 + n_2^*)$$

where n_2^* is the number of conceptually unique input and output parameters for invoking this algorithm. A minimal implementation of any algorithm implies invoking the algorithm by specifying just its name and parameters instead of writing it.

Any program with volume V is considered to be implemented at a program level L, defined by

$$L = V^*/V$$

Halstead called the inverse of the program level as the difficulty D.

$$D = 1/L$$

Since L is difficult to compute as n_2^* is not easily found for all the algorithms, Halstead proposed an alternate method to estimate the program level. This is given by

$$L_{est} = (2/n_1) * (n_2/N_2)$$

The effort E in software science is defined as

$$E = V * D_{est}$$

The unit of measurement of E is number of elementary mental discriminations which a human mind can make. This terminology is borrowed from the field of cognitive psychology.

Halstead also hypothesized that the programming time T of a program in seconds can be defined as

$$T = E/b$$

where b is the Stroud Number [20]. This Stroud number has a range of 5 to 20 but is set rather arbitrarily to 18 by Halstead.

Finally the language level LL of a programming language is defined as

$$LL = L * V^* = L^2 * V$$

1.1.3 McCabe's Metrics

T. J. McCabe [31] proposed a metric to correspond to the difficulty programmers experience in maintaining a program. Since the number of different independent paths through a program adds to the complexity of testing a program, he proposed a metric based on the number of conditions in the program. This metric is a control flow complexity metric called cyclomatic complexity.

The cyclomatic complexity is defined as

$$v(G) = e - n + p$$

where e is the number of edges, n is the number of nodes and p is the number of connected components in the control flow graph of the program. For this measure a branch from the exit node to the entry node is added before its computation. Therefore, for a connected control flow graph with e edges and n nodes,

$$v(G) = e - n + 2$$

It can be shown that an equivalent formula for v(G) can be given as [31]

$$v(G) = DE + 1$$

where DE is the number of predicates in the program. v(G) can be easily calculated using this alternate form.

1.1.4 Residual Complexity Metric

A new metric proposed by M. H. Samadzadeh and W. R. Edwards, Jr., is called the residual complexity metric [36]. The term Residual implies the remaining complexity in a software document after some attempt has been made to understand it either in whole or in parts by conceptual subdivision or chunking.

Samadzadeh and Edwards contend that understanding a software document can be modeled by a token categorization process. A software document can be thought of as consisting of tokens which belong to different token classes. These token classes need not be necessarily disjoint and hence, in general, they form a covering. A person trying to understand a software document, obtains the tokens from the document and finds the token class to which each token belongs. After all the tokens are classified, the portion of the complexity still uncovered can be represented as [36]

$$R = N_1 * \log(N_1) + N_2 * \log(N_2) + \ldots + N_q * \log(N_q)$$

where R is called the residual complexity metric and N_i is the number of tokens in the ith class or block of the current partition.

1.2 Objectives

This research effort could be termed an experiment in software science. The following are the major objectives of this work:

1. To present a brief introduction to the field of software metrics and definition of some of the more popular software metrics.

2. To discuss a few of the experiments and studies conducted to validate and analyze software metrics.

3. To analyse C software in a small software house through collection of static metrics and their

interrelationships.

4. To explore the relationship between static code metrics and fault rates in the available application area.

Chapter II reviews the major studies in software complexity measures. Chapter III presents the data collection methodology for this paper and the results thereof. The results are summarized in Chapter IV and suggestions are also given for future studies in this area.

CHAPTER II

A SURVEY OF EXPERIMENTS IN SOFTWARE METRICS

There has been continued interest in the study of software metrics with Halstead's and McCabe's metrics being the more popular ones. There have been measures proposed to evaluate the software complexity measures themselves [41], ideas to share industrial software data in an encrypted form to avoid security problems [21,23] and numerous validation studies for Halstead's and McCabe's metrics across programming languages and different application areas [5,10,12,15,17,24,30,43]. In addition, there have been extensions of both measures [2,7,11,39], attempts to combine the characteristics of both measures [35], analysis of software defect models [42], suggestions to predict programmer performance based on software metrics [13], etc. It should also be noted that questions have been raised about the basis of Halstead's measures, the most notable being [9], where Coulter asserts that Halstead incorrectly applied some results from cognitive psychology as part of his assumptions.

Included in this section are descriptions of several studies of the effort involved in the programming process.

Models, experiments and results are presented for a wide range of studies.

2.1 Experiments Involving Halstead's Metrics

Halstead [18,19,20] was instrumental in paving the way for a new field of computer science called Software Science and his metrics have been a source of continued research to this date. Halstead designed two series of experiments to validate his model to predict programming time. These are described below.

According to Halstead, the time in seconds to produce a program (HT) can be expressed as

$$HT = (n_1 * N_2 * N * \log(n)) / (2 * n_2 * b)$$

(For an explanation of the symbols used, please see Section 1.1.2.)

In the first experimental series [19], Halstead selected a graduate student doing Ph.D. thesis under his direction, as his subject. Together they selected 12 algorithms published in the Communications of the ACM in the early 1960's. The subject implemented each of the algorithm in three different languages: PL/I, FORTRAN and APL while maintaining a high degree of mental concentration. In order to avoid learning effects, the order of the implementation for each algorithm was randomized. The subject accurately recorded the time for each implementation including the time for desk-checking. The results are given in Table 1. In this table, T represents the actual implementation time and HT represents time predicted by Halstead's estimator, and the CACM number gives the number of the algorithm that appeared in Communications of the ACM in the 1960's.

As seen from the table, this experiment resulted in a correlation coefficient of 0.92 between T and HT. Thus the Halstead estimator is seen to account for nearly 85% of the variance in the observed implementation time and performs remarkably, based on this experiment alone. Since programming process involves human factors, this result must be confirmed by independent studies.

Table 1

PREDICTION OF T(ACTUAL TIME) USING HALSTEAD'S MODEL(HT)

CACM Number	First		Second		Third	
	T	HT	T	HT	T	HT
(14)	33	13	15	16	25	29
(16)	135	123	77	102	44	53
(17)	33	22	10	6	11	10
(19)	7	3	10	9	9	5
(20)	12	9	14	7	6	5
(21)	43	51	30	47	39	63
(23)	21	17	13	5	13	3
(24)	16	22	8	7	6	6
(25)	62	101	45	74	20	42
(29)	25	5	35	9	16	17
(31)	20	17	11	15	7	8
(33)	4	1	3	1	3	1
Correlation		.92		.92		.94

(Source: M. H. Halstead, Elements of Software Science, Elsevier, New York, 1977, p. 53)

Conte et al. [8], attempted to derive a simple time model based on the size metric alone. Based on the size metrics provided along with the results of some experiments by Halstead himself [19], they derived the following regression models for the three implementations,

First: $T_N(est) = -7.4 + 0.25 * N$

Second: $T_N(est) = -2.1 + 0.15 * N$

Third: $T_N(est) = 1.3 + 0.096 * N$

where T_N(est) is the estimator based on size and the correlations are respectively 0.97, 0.93 and 0.86 between T_N(est) and T. There is not much difference between these regression models and Halstead's Time Equation.

In the second experimental series Halstead set out to confirm his previous results. He had another graduate student select 11 algorithms from textbooks. This time the algorithms were implemented in FORTRAN alone and the times recorded included time for reading specification, designing, coding and testing. The results are given in Table 2. This time the correlation between T and HT was 0.82.

Table 2

PREDICTION OF T(ACTUAL TIME) USING SIZE OR HALSTEAD'S MODEL(HT)

Program Number	T(mins)	LOC	N	HT
1	5	7	53	5
2	5	8	50	5
3	21	11	73	2
4	30	15	131	7
5	16	18	113	16
6	19	18	110	15
7	24	18	128	23
8	39	32	244	44
9	92	36	428	82
10	43	38	228	49
11	91	59	483	129
Correlation		.89	.94	.82

(Source: M. H. Halstead, <u>Elements of Software Science</u>, Elsevier, New York, 1977, p. 56)

Halstead's work showed the possibility of applying a rigorous scientific approach to the programming process. Even though simple models based on size metric have a high correlation with implementation time, Halstead's Time Estimator Model is preferable as it does not use regression derived constants.

2.2 Basili's Metric Analysis

The Software Engineering Laboratory (SEL) is a joint venture involving the University of Maryland, NASA Goddard Space Flight Center, and the Computer Sciences Corporation. The laboratory collects data from commercial programs that deal with many aspects of software development process. Basili et al. [5] conducted an empirical investigation on the data collected by SEL, and their results are summarized below.

The programs analyzed in [5] are the newly developed modules of ground support software for satellites. The parent systems consist of 51,000-112,000 lines of FORTRAN code (including comments) in 200-600 modules. They were produced by 8-23 programmers who spent 6,900-22,300 programmer-hours on their development.

In order to improve the accuracy of the data collected, Basili et al. used various selection criteria for the data to be analyzed. Each programmer completed a Component Status Report (CSR) each week. The programmer reported effort spent on each module, for designing, coding testing and documentation. Since the programmers may miss recording information, the managers were asked to complete a Resource Summary Form (RSF) every week, reporting time charged for various personnel. Thus one of the criteria to select the projects was

$$Vm = \frac{\text{Number of CSRs by Programmers}}{\text{Number of weeks Programmer appears in RSFs}}$$

A higher value of Vm (maximum of 1) indicates a greater reporting frequency and forms a good selection criterion.

Table 3

RANK CORRELATIONS OF T(ACTUAL TIME) WITH VARIOUS METRICS FOR BASILI'S STUDY

Vm(#modules)	Based on:	LOC	N	v(G)	HT
All(731)		.46	.43	.46	.45
80%(398)		.42	.42	.46	.42
90%(215)		.44	.46	.48	.47

(Source: V. R. Basili, R. W. Selby, Jr., and T. Philips, "Metric Analysis and Data Validation Across Fortran Projects," IEEE Trans. on Software Eng., Vol. SE-9, No. 6, pp. 653-663, Nov. 1983.)

2.2.1 Analysis

The correlation coefficients between the total development effort and the various metrics, reported by Basili et al. [5], are listed in Table 3. These coefficients are significant at the 0.001 level, and range from 0.42 to 0.48. Thus it is seen that each of the metrics LOC, N, v(G) and HT account for nearly 25% of the variation in the actual error finding time, all independently.

2.3 Curtis' Maintenance Analysis

Curtis, Sheppard and Milliman [12] wanted to correlate the complexity metrics with the time it takes to locate a bug purposely planted in a program. They designed an experiment in which the subjects were provided the program listing, input files, desired output, and the erroneous output after the bug was inserted. The subjects were asked to locate the bug and the time to do this was measured to the nearest minute. The complexity metrics that were correlated with this time, included lines of code, McCabe's Cyclomatic Complexity and Halstead's Effort metric.

Fifty four professional programmers, averaging 6.6 years of programming experience, were chosen as the subjects for this experiment. Three types of programs, namely sorting, accounting and grading, were selected as test programs. Each of the programs was developed with three different lengths varying from 25 to 225 LOC. In addition three types of control flow were used, structured, unstructured and natural control flows, giving rise to 27 programs. Then three types of bugs, computational, logical and data bugs, were introduced for each of the 27 programs producing 81 programs in all. The subjects were asked to read the program and identify the bug. They were not asked to fix the bug. The time to read the program and identify the bug was noted accurately.

Table 4

CORRELATIONS OF METRICS WITH TIME TO FIND A BUG

Source	Number	E	v(G)	LOC
Subroutine	27	.66	.63	.67
Program	27	.75	.65	.52

(Source: B. Curtis, S. B. Sheppard, P. Milliman, M. A. Borst and T. Love, "Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics," IEEE Trans. on Software Eng., Vol. SE-5, No. 2, pp. 96-104, March 1979.)

Table 4 summarizes the correlation coefficients relating the time to locate the bug and the complexity metrics. The first row in the table corresponds to the metrics measured at the subroutine level, and the second row to the metrics measured at the program level. When the metrics were analyzed for intercorrelations it was found that all of them were highly correlated at the subroutine level but the correlations between E with either v(G) or S_s was substantially lower than of v(G) and S_s at the program level.

This study suggests that complexity metrics such as E and v(G) are predictive of the effort involved in finding errors and can be used by project managers while allocating resources for software maintenance.

2.4 Crawford's Fault Analysis

This study has more relevance to the present paper than the earlies studies presented. The system analyzed in [10] consists of 89 units containing 2040 files with 516K deliverable source lines or 258K non-commentary source lines of C code. The metrics collected in this study were Size metrics, Halstead measures and McCabe's metrics. The issues addressed in this paper are intermetric relationships and the correlation of complexity metrics with software fault rates. Table 5 lists the correlations between metrics reported in this study. (HN is the Halstead estimator for N).

Table 5

TABLE OF RELATIONS AMONG STATIC METRICS
IN CRAWFORD'S STUDY

Variables	Raw Data	Log Transform of Data
size,N	.971	.989
size,V	.977	.990
size,E	.901	.975
size,v(G)	.935	.952
N,HN	.928	.962
V,E	.874	.402

(Source: S. G. Crawford, A. A. McIntosh and D. Pregibon, "An Analysis of Static Metrics and Faults in C Software," The Journal of Systems and Software, Vol. 5, pp. 37-48, 1985)

Crawford, McIntosh and Pregibon's results confirm that all metrics correlate to a fair degree with program size and with each other. On another part of their study, a model was developed for the fault rate in terms of the static metrics. The dependent variables defined for this purpose are:

CF: Code-faults, which are faults attributable to the detailed coding of a unit.

ETI: Effort-to-isolate, the total estimated development effort to isolate the CFs in a unit.

ETR: Effort-to-repair, the total estimated effort to fix each CF in a unit, retest the unit, and deliver the change for system integration.

The basic metrics used in this model were, non-commentary source lines (NCSL), Cyclomatic Complexity (VGC), the total delivered source lines (DSL) and the total number of functions defined (FND).

The models for code faults, isolation and repair efforts were proposed as,

$$\ln(CF) = 3.2 + 0.32 * \ln(NCSL) + 0.54 * \ln(DEC) - 3.0 * \ln(DEN) + 1.1 * \ln(DOC) + 0.92 * VA1$$

$$\ln(ETI) = 3.0 + 0.29 * \ln(NCSL) + 0.71 * \ln(DEC) - 3.4 * \ln(DEN) + 1.1 * \ln(DOC) + 3.2 * VA2$$

$$\ln(ETR) = 4.1 + 0.18 * \ln(NCSL) + 0.88 * \ln(DEC) - 3.4 * \ln(DEN) + 1.6 * \ln(DOC) + 3.7 * VA2$$

where the derived metrics used in this model are defined by:

DEC: number of decisions per function defined, VGD/FND

DEN: number of tokens per non-commentary source line, (N1 + N2)/NCSL

DOC: ratio of non-commentary to total source lines, NCSL/DSL

VA1 and VA2 are the operator and operand counts adjusted by NCSL.

VA1: $\ln(n_1) - [2.6 + 0.16 * \ln(NCSL)]$

VA2: $\ln(n_2) - [0.88 + 0.75 * \ln(NCSL)]$

and ln is the natural logarithm.

These derived metrics represent decisions per function (DEC), token density (DEN), the lack of block comment documentation (DOC), and vocabulary (VA1,VA2) of a software unit. The squared multiple correlation coefficients for these models are respectively 0.53, 0.48, and 0.46. Thus the variables NCSL, DEN, DOC, DEC and VA2 account for 48% of the variability in isolation effort (ETI).

The models proposed by Crawford et al., account for nearly half the variation in Code Faults, Effort to Isolate, etc.. If this could be improved upon, the models could be used in allocating resources for each of the different aspects of software maintenance, namely testing, identifying errors, isolating errors and repair.

CHAPTER III

SOFTWARE ANALYSIS

This section describes the software project analyzed, the data collection methodology, the static metrics gathered, the analysis of the static metrics, comparison of results to previous findings and the model derived from the data analyzed for code faults.

3.1 Experimental Methodology

The experiment in this study involved a software product written by professional programmers from by a small software house. To characterize the methodology, a comprehensive and well known frame of reference is the classification scheme for Software Engineering experiments suggested by V. R. Basili et al. [3]. According to their classification, this study is designed to understand (motivation) about the software (object) to characterize (purpose) from the view of the maintainer (perspective) the program/project (domain) for a single-project (scope). The experiment is designed on multivariate regression analysis (design), to reflect directly cost of errors and changes (criteria) by gathering metrics objectively (measurement). The study analyzes the work of a small group of programmers, professional programmers with an experience of approximately 3 years and with similar backgrounds. Hence the limitations in extrapolating the results presented, to the entire programming community must be evaluated as described in [6]. It should also be noted that control experiments involving programming like the experiment conducted by Mitchell et al. [33], come under a lot of criticism regarding the generalization and interpretation of their results.

3.2 Software Project Characteristics

The software project in this study was a medium-sized database system in its maintenance phase. There were 569 files with 180K source lines. The source code consisted of used and new code for the latest release. The data was collected on an as-is-where-is basis. This implies that no transformation was done to any part of the code before it was subjected to analysis. The software is decomposed in the following fashion based on the convention used by Crawford et al. [10]:

subsystem: a major functional entity;

unit: a basic development entity, separately tracked, and functionally decomposed from the subsystem;

file: a separately compilable entity;

function: the basic callable entity implementing a software routine.

For example, the file tupsup.c is found in the directory 'tuplepro\support'. All files under 'tuplepro' directory form a subsystem, all files under 'support' form a unit and a function 'snapshot' in the above file forms the basic callable entity.

The error data was collected from the information available from the header of each file. This error data had been manually entered into the header section by the programmer who had made that change, and was done before integrating the error fixes into the system. Since this practice had been followed consistently, the error data provided a fairly accurate source of information to be correlated with the static metrics. The data was collected from a source code management system [34] which maintains all the versions and the revisions of the software in a compact form. Since any modification to a file, either for enhancement or for error fix had been duly recorded in the header section of the file, differences between successive versions of a file were retrieved from the source code management system.

The file level is the level in the software decomposition for which fault data was collected.

Correspondingly, the file level was where code measurements were made. As an abstraction each file was considered a module. Files are the basic entities tracked by the source code management system, and a file is the work of a single person. Analysis done at this level may be used to improve software control practices. The file development could be monitored based on either a size metric or McCabe's metric. A threshold measure for a metric could be established; and while developing a file, if the metric value crosses the threshold value, then the file could be split into two files, easing their maintainability.

C was the primary source language used in the software system [28]. Static analysis tools were developed exclusively to measure characteristics of C source code, so the small part of the software system employing assembly code (approximately 2%) was excluded from the analysis.

The complete listing of all the modules and their associated metrics is included in Appendix A. In this listing AUTH stands for a single letter author identification code, VER for version, LOC for lines of code, BL for number of blank lines, SC for number of semicolons, VGC for McCabe's cyclomatic complexity, N1 for the number of unique operators, N2 for the number of unique operands, N1TOT for the total number of operators, N2TOT for the total number of operands, HN for length in tokens and HNEST for estimated length. HV is the volume, HE is the effort measure, R1, R2, R3, R4 are the residual complexities based on total occurrences of tokens and R1U, R2U, R3U, and R4U are the Residual Complexities based on unique occurrences of tokens. Section 1.1.4 contains a discussion of the residual complexity metric.

3.3 Size Metrics

Section 1.1.1 presents many possible ways to quantify the size of a software system. For consistency, the following definitions of size metrics based on Crawford et al. [10], were used.

DSL: delivered source lines, a count of all lines of code delivered and necessary to build the software system;

NCSL: non-commentary source lines, a count of developed lines after comments and blank lines have been filtered out;

SC: a count of semicolons (the C statement terminator); a rough measure of the number of statements or 'clauses'.

For DSL, NCSL and SC, the code was measured before the use of any preprocessors to expand macros or include files.

This is consistent with the practice followed by Crawford et al. [10]. The relations among these metrics are illustrated in Appendix B. Each plot is a log-log scatter plot of one size measure against another. In these plots, LOG_SC is log(SC), LOG_NCSL is log(NCSL) and LOG_DSL is log(DSL). The plots show the overall linear relations between these measures.

Table 6 depicts the frequency distribution of NCSL. It can be seen that there were a large number of small modules and a small number of large modules. The mean NCSL was approximately 320 LOC.

Table 6

FREQUENCY DISTRIBUTION OF NCSL

Range	#modules
0 -40	255
40 -120	165
120-200	53
200-280	38
280-360	14
360-440	7
440-520	7
520-600	7
600-680	7
680-760	14

DSL, NCSL and SC are highly correlated with each other. Thus any of these measures could be selected to represent the size metric. NCSL was chosen to represent the size measure because it is independent of both the consistency and use of comments, and it can be applied to all languages relatively unambiguously.

3.4 Static Code Metrics

The relationship between static code metrics and fault rates in the available environment aid in the development of a model for the fault rate based on static metrics. The metrics were collected at the file level. The metrics were gathered from the code as the programmer had written it, before either other files were included or macros were expanded.

A series of tools was developed for analysis of C source code. These data gathering tools are included in Appendixes C, D, and E. The implementation uses algorithms for parsing and identifying tokens. The metrics were chosen because they are currently popular in software engineering literature and are also easily computable.

3.5 McCabe's Metric

McCabe's control flow complexity metric (cyclomatic complexity) VGC, based on the number of conditions in a program was included in this study [31]. This metric can be simply given as,

VGC: number of conditions + 1

A compound decision such as if (C1 and C2) is equivalent to if (c1) then if (C2). Such a compound decision counts as two conditions in the metric given above.

This metric was originally designed to measure the number of "linearly independent" paths through a program, which in turn is believed to relate to the testability and maintainability of the program [31]. Constructing a software tool that generates the flow graph of a program for analysis is complicated. But an algorithm to compute VGC based on the equation given above is computationally tractable. A software analyzer was written to compute VGC by finding the number of decisions. This analyzer is included in Appendix D.

3.6 Halstead's Metrics

Halstead's four basic counts [19] were used.

n1: number of unique operators

n2: number of unique operands

N1: number of total operators

N2: number of total operands

As seen earlier in Section 1.1.2, there is no general agreement on exactly which tokens are operators and which tokens are operands. It was decided to implement the counting rules for C, in a way that would be consistent with rules developed for Pascal as distributed by the Purdue University Software Metrics Research Group and described in [8]. The counting rules classify all tokens as either operators or operands, most user-defined tokens as operands, and since C permits repetitive tokens (i++ representing i=i+1) sometimes multiple tokens are counted as one distinct operator (++ is a single operator).

The other Halstead metrics that were gathered are:

Hn: Vocabulary

$$n = n_1 + n_2$$

HN: Program length

$$N = N_1 + N_2$$

HN_{est}: Estimated program length

$$N_{est} = n_1 * \log(n_1) + n_2 * \log(n_2)$$

HV: Program Volume

$$V = N * \log(n)$$

HD: Difficulty

$$D = (n_1/2) * (n_2/N_2)$$

HE: Effort

$$E = V * D$$

The software analyzer to collect Halstead's metrics is listed in Appendix E. This analyzer uses a regular automaton to parse the tokens. Once the tokens are parsed, they are simply classified as either operators or operands. From the counts of operators and operands, all Halstead's metrics are computed.

3.7 Residual Complexity Metrics

These metrics are derived from the basic tokens obtained as either operators or operands and subjecting them to further classification. Section 1.1.4 gives the definition of residual complexity. The four classification schemes considered in this study are,

1. Operator, Operand tokens
2. Sequential, Conditional, and Repetitive tokens
3. Input, Output, and Processing tokens
4. (Arithmetic, logical, system) operators and (Constant, Variable) operands

It should be noted that classification 1 is Halstead's operator-operand classification and classification 4 is a refinement of classification 1. As noted in Section 1.1.4, the formula for the residual complexity, after classifying the tokens into 'q' equivalence classes is,

$$R = N_1 * \log(N_1) + N_2 * \log(N_2) + \ldots + N_q * \log(N_q)$$

where N_i is the number of tokens in the ith set, for $1 \leq i \leq q$. For this study two definitions of N_i were used. One was the count of the number of unique tokens in equivalence class i, and the other was the count of total occurrences of tokens in class i. This gives rise to 8 sets of measures, 2 measures for each of the 4 classification schemes. Thus, for the jth ($1 \leq j \leq 4$) classification scheme, we have two definitions of residual complexity metric namely,

$$R_j \text{ and } R_{juniq}$$

where the former is defined in terms of total occurrences of tokens, and the latter is defined in terms of unique tokens.

The tools developed to collect Halstead' metrics for C source code were extended to gather these metrics, by subjecting the tokens collected to the different classification schemes. The initial classification is classification one; i.e., R_{1uniq} is HN_{est} and R_1 is simply,

$$N_1 * \log(N_1) + N_2 * \log(N_2)$$

3.8 Analysis of Static Metrics

The following subsections discuss the methodology used for the analysis of the static code metrics. Plots are used liberally to make the discussion clearer. All data analysis and graphical displays were done using the SAS package [22].

A logarithmic transform of the basic data is used in most of the analysis. This is done in order to compress the wide range of metric values to a normalized range.

The intermetric correlations are included in Appendix B and a portion of it is reproduced in Table 7. It is seen that all the static metrics are highly correlated with each other and in particular with the size metric. Comparing this result with the result of Crawford et al., [10] the interrelationships are consistent with historical data.

Table 7

INTERMETRIC CORRELATIONS

	NCSL	HE	VGC	R1	R2
NCSL		.93	.96	.98	.98
HE			.87	.97	.97
VGC				.86	.93
R1					.99

3.9 Subjective Ratings

A survey was conducted to correlate the perceived complexity to all the static measures in general, and to the residual complexity metrics in particular. The survey was conducted while the software was in its maintenance phase. The programmers in charge of software maintenance were requested to rate different program modules according to their perceived complexity. Since the original developers are different from the people who are maintaining the software, the possibility of a programmer rating his/her own program was avoided. A six-point scale was used to record the ratings.

PC: perceived complexity of the file module (1=least complex, 6=most complex)

The programmers were asked to rate only those modules, they had worked with and were familiar with.

3.10 Analysis of Subjective Ratings

The subjective rating was correlated with the following measures: VGC, HE, R_j, and R_{juniq} for $1 \leq j \leq 4$. The correlations are tabulated in Table 8. It can be seen that the perceived complexity is strongly correlated with VGC. It is also observed that R1, R2, R3, R4 correlate to a fair degree with PC and perform significantly better than Halstead's Effort measure in correlating with subjective complexity. The hypothesis that the residual complexities based on total occurrences of tokens correlate to the perceived complexity better than those based on unique occurrences of tokens was rejected at a sigificance level of

0.1 though the former have better correlation coefficients than the latter.

Table 8

CORRELATION OF SUBJECTIVE COMPLEXITY WITH METRICS

Metric	Explanation	Correlation
NCSL	Non commentary Lines	.65
VGC	McCabe's Measure	.67
HE	Halstead Effort Metric	.57
R1 R2 R3 R4	Residual Complexities Based on Total Occurrences of Tokens	.60 .60 .59 .60
R1U R2U R3U R4U	Residual Complexities Based on Unique Occurrences of Tokens	.52 .52 .57 .52

3.11 Fault Data

A software fault is defined as "an accidental condition that causes [an entity of hardware, software, or both] to fail to perform its required function" [25]. The number of errors attributed to each file was gathered from the source code management system. It was desired to relate the static measures of source code to software faults. In the life cycle of a software document, static code metrics can be collected only during its implementation stage, and thus they can be used to model code faults (CFs) which are attributable to this stage.

When the fault data was analyzed it was found that each of the software metrics explains approximately one-quarter of its variation and no one measure does significantly better than the others. An attempt was made to find an optimal model for CF by including all possible combinations of measures by running the Step-Wise Linear Regression Model of SAS package [22], and the model is presented below,

$$CF = -0.35 + 0.96 * \ln(HN) - 1.2 * \ln(NCSL) + 0.27 * \ln(VGC) + 0.03 * R1U - 0.01 * R2U$$

$$- 0.1 * R4U - 0.001 * R4 + 0.001 * R2$$

The correlation coefficients and the regression models are given in Appendices B.12 and B.14, respectively. The model given above should provide a reasonable approximation to the fault process. Since the independent variables in this model can be extracted using the software tools listed in Appendices C, D and E, the analysis of the entire software system could be automated easily. There is little overhead once the basic software tools to gather these metrics are either developed or purchased. A project management team may evaluate the applicability of this model to the project at hand, and if the module is found suitable, may use this model to identify error-prone modules and allocate resources to maintain them.

3.12 Analysis Across Versions

An important distinction of this study from similar studies is the analysis of a software product across its version changes as related to the corresponding static metric changes. The correlations between the static measures for each version is included in Appendix B. In these listings version 1.0 is the base version. The static measures that are included are NCSL, HN, HE, VGC, R1, R2, R3, R4, R1U, R2U, R3U and R4U. The successive versions and the number of modules in each version (within brackets) are 1.0(156), 1.1(21), 1.11(3), 1.12(2), 1.2(17), 1.3(9), 1.4(250), 1.5(33), 1.6(51), 1.7(11), 1.8(9) and 1.9(7). For example, the correlation coefficients between VGC and R1 are 0.88, 0.90, 0.98, 1.0, 0.87, 0.92, 0.92, 0.92, 0.95, 0.98, 0.97, 0.90 respectively for versions 1.0, 1.1, 1.11, 1.12, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9. Thus VGC and R1 are uniformly related throughout all the versions as are other different pairs of static metrics. Looking through the data, it can be perceived that as the software evolves through its versions the uniformity in interrelations between the static measures remain consistent. This is surprising because the software is usually modified extensively by a person different from one who originally wrote it.

CHAPTER IV

SUMMARY, CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

A model for code fault has been presented. It is found that each of the following metrics NCSL, VGC, HE, R_j and R_{juniq} $(1 \le j \le 4)$ explains approximately one-quarter of the variance in code faults. The interrelationship between static code metrics was investigated and found to be consistent with similar studies. The static code metrics were found to correlate strongly with each other.

The relationship among the static metrics across versions was investigated. It was found that the relationships were fairly consistent through different versions and hence any software model which is based on strong correlation between certain metrics, could be applied at any stage of software maintenance.

In this study, the validation of residual complexity metric has been extended. This metric symbolizes the human understanding process of a software document to a reasonable degree of accuracy. Among the four classifications investigated, the first classification; namely, classifying the tokens into operators and operands, which is nothing but the Halstead classification, seemed to correspond closely to the perceived complexity.

Future studies may address the issue of extending the classification scheme presented in this work. Also as only one-quarter of the variance in the code faults has been quantified, it may be worthwhile to try to quantify as much of the remaining variance as possible. Finally, the practicality of the model presented, could be empirically investigated. This might require both the monitoring of modules of a project for a significant period of time for their code fault rate, and the correlating of this data with the rate predicted by this model.

REFERENCES

[1] A. J. Albrecht, "Measuring Application Development Productivity," Proc. of the Joint SHARE/GUIDE Symposium, pp. 83-92, Oct. 1979.

[2] A. L. Baker and S. H. Zweben, "A Comparison of Measures of Control Flow Complexity," Proc. of IEEE COMPSAC, pp. 695-701, 1979.

[3] V. R. Basili, R. W. Selby and D. H. Hutchens, "Experimentation in Software Engineering", IEEE Trans. on Software Eng., Vol. SE-12, No. 7, pp. 733-743, July 1986.

[4] V. R. Basili and B. T. Perricone, "Software Errors and Complexity: An Empirical Investigation," Comm. of ACM, Vol. 27, No. 1, pp. 42-52, Jan. 1984.

[5] V. R. Basili, R. W. Selby, Jr. and T. Philips, "Metric Analysis and Data Validation Across Fortran Projects," IEEE Trans. on Software Eng., Vol SE-9, No. 6, pp. 653-663, Nov. 1983.

[6] R. E. Brooks, "Studying Programmer Behaviour Experimentally: The Problems of Proper Methodology", Comm. of ACM, Vol. 23, No. 4, pp. 207-213, 1980.

[7] G. Cantone, et al., "Complexity in Program Schemes: The Characteristic Polynomial," ACM SIGPLAN Notices, Vol. 18, No. 3, pp. 22-30, Mar. 1983.

[8] S. D. Conte, H. E. Dunsmore and V. Y. Shen, Software Engineering Metrics and Models, The Benjamin/Cummings Publishing Co., Menlo Park, California, 1986.

[9] N. S. Coulter, "Software Science and Cognitive Psychology," IEEE Trans. on Software Eng., Vol SE-9, No. 2, pp. 167-171, March 1983.

[10] S. G. Crawford, A. A. McIntosh and D. Pregibon, "An Analysis of Static Metrics and Faults in C Software," The Journal of Systems and Software, Vol. 5, pp. 37-48, 1985.

[11] K. Culik, "The Cyclomatic Number and the Normal Number of Programs," ACM SIGPLAN Notices, Vol. 14, No. 4, pp. 12-17, Apr. 1979.

[12] B. Curtis, S. B. Sheppard, P. Milliman, M. A. Borst and T. Love, "Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics," IEEE Trans. on Software Eng., Vol. SE-5, No. 2, pp. 96-104, March 1979.

[13] B. Curtis. et al., "Stronger Prediction of Programmer Performance by Software Complexity Metrics," Proc of IEEE's Fourth Int. Software Eng. Conference, pp. 356-360, 1979.

[14] H. E. Dunsmore and J. D. Gannon, "Data Referencing: An Empirical Investigation," IEEE Computer, Vol. 12, No. 12, pp. 50-59, Dec. 1979.

[15] J. L. Elshoff, "An Analysis of Some Commercial PL/I Programs," IEEE Trans. on Software Eng., pp. 113-120, June 1976.

[16] A. R. Feuer and E. B. Fowlker, "Some Results from an Empirical Study of Computer Software," Proc. of IEEE Fourth Int. Conference on Software Eng., pp. 351-355, 1979.

[17] J. D. Gannon, E. E. Katz and V. R. Basili, "Metrics for ADA Packages: An Initial Study," Comm. of ACM, Vol. 29, No. 7, pp. 616-623, July 1986.

[18] M. H. Halstead, "Advances in Software Science", Advances in Computers, Vol. 18, Academic Press Inc., New York, pp. 119-172, 1979.

[19] M. H. Halstead, Elements of Software Science, Elsevier, Ed. Yovits, New York, 1977.

[20] M. H. Halstead, "Natural Laws Controlling Algorithm Structure," ACM SIGPLAN Notices, pp. 19-26, Feb. 1972.

[21] W. Harrison and Cook, "A Method of Sharing Industrial Data," ACM SIGPLAN Notices, Vol. 20, No. 2, pp. 42-53, Feb. 1985.

[22] J. T. Helwig, SAS Introductory Guide, Kingsport Press, Chapel Hill, North Carolina, May 1978.

[23] S. Henry, "A Technique for Hiding Proprietary Details While Providing Sufficient Information for Researchers", The Journal of Systems and Software, Vol. 8, pp. 3-11, Jan. 1988.

[24] S. Henry and D. Kafura, "Software Structure Metrics Based on Information Flow," IEEE Trans. on Software Eng., Vol. SE-7, No. 5, pp. 510-518, Sep. 1981.

[25] IEEE Standard Glossary of Software Eng. Terminology, IEEE, New York, Feb. 1983.

[26] D. Kafura and S. Henry, "Software Quality Metrics Based on Interconnectivity," The Journal of Systems and Software, Vol. 2, pp. 120-131, 1981.

[27] K. M. Kavi and U. B. Jackson, "Effect of Declarations on Software Metrics: An Experiment in Software Science," ACM Sigmetrics, Vol. 11, No. 2, pp. 57-71, Summer 1982.

[28] B. W. Kernighan and D. M. Ritchie, The C Programming Language, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1978.

[29] J. L. Lassez and D. V. D. Knijff, "Evaluation of Length and Level for Simple Program Schemes," Proc. of IEEE COMPSAC, pp. 688-694, 1979.

[30] H. F. Li and W. K. Cheung, "An Empirical Study of Software Metrics", IEEE Trans. on Software Eng., Vol. SE-13, No. 6, pp. 697-708, June 1987.

[31] T. J. McCabe, "A Complexity Measure," IEEE Trans. on Software Eng., Vol. SE-2, pp. 308-320, Dec. 1976.

[32] C. L. McClure, "A Model for Program Complexity Analysis," Proc. of IEEE's Third Int. Software Eng. Conference, pp. 149-157, 1978.

[33] J. Mitchell, J. E. Urban and R. McDonald, "The Effect of Abstract Data Types on Program Development", Computer, pp. 85-88, Aug. 1987.

[34] PVCS User's Manual, Polytron Corporation, Beaverton, Oregon, Jan. 1988.

[35] B. Ramamurthy and A. Melton, "A Synthesis of Software Science Measures and Cyclomatic Number," IEEE Trans. on Software Eng., Vol. 14, No. 8, pp. 1111-1116, Aug. 1988.

[36] M. H. Samadzadeh and W. R. Edwards, Jr., A Classification Model of Software Comprehension, Dept. of

Computer Science, Oklahoma State University, OSU-CIS-TR-88-01, 30 pages, 1988.

[37] N. F. Schneidewind and H. Huffman, "A Software Science Analysis of COBOL Programs," CSD-TR-348, Dept. of Computer Science, Purdue Univ., Aug. 1980.

[38] S. M. Shatz, "Towards Complexity Metrics for ADA Tasking," IEEE Trans. on Software Eng., Vol. 14, No. 8, pp. 1122-1128, Aug. 1988.

[39] J. J. Tamine, "On the Use of Tree-like Structures to Simplify Measures of Complexity," ACM SIGPLAN Notices, Vol. 18, No. 3, pp. 62-68, Sep. 1983.

[40] D. M. Weiss and V. R. Basili, "Evaluating Software Engineering Analysis Changes: Some Data from the Software Engineering Laboratory," IEEE Trans on Software Eng., Vol SE-11, No. 2, pp. 157-168, Feb 1985.

[41] E. J. Weyuker, "Evaluating Software Complexity Measures," IEEE Trans. on Software Eng., Vol. 14, No. 9, pp. 1357-1365, Sep. 1988.

[42] T. J. Yu, V. Y. Shen and H. E. Dunsmore, "An Analysis od Several Software Defect Models," IEEE Trans. on Software Eng., Vol. SE-14, No. 9, pp. 1261-1270, Sep. 1988.

[43] S. H. Zweben and K. C. Fung, "Exploring Software Science Relations in COBOL and APL," Proc. of IEEE COMPSAC, pp. 702-707, 1979.

[44] J. C. Zolnowski and D. B. Simmons, "Taking the Measure of Program Complexity," Proc. of National Computer Conference, pp. 329-336, 1981.

APPENDIXES

APPENDIX A

MODULES AND METRICS LISTING

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
1	ADOCNUM.C	G	1.00	74	18	36	47	13	17	25	195	176	371	185.6
2	ANDEXPR.C	H	1.40	16	10	16	7	5	14	4	37	26	63	106.6
3	ANDNOTEX.C	H	1.40	16	9	16	7	6	14	14	39	28	67	106.6
4	ANDNOTOP.C	A	1.40	83	26	51	42	12	22	27	202	133	335	226.5
5	ANDNOTOP.C	A	1.60	101	28	59	54	14	29	42	265	173	438	367.4
6	ANDOP.C	U	1.40	105	29	64	46	15	22	27	242	162	404	226.5
7	ANDOP.C	U	1.60	117	32	72	54	16	29	42	283	187	470	367.4
8	AREADDR.C	G	1.00	33	21	27	12	4	24	48	129	98	227	378.1
9	AREADV.C	G	1.00	15	13	19	9	4	12	20	46	41	87	129.5
10	ASFE.C	U	1.40	280	129	80	184	45	32	140	776	630	1406	1158.1
11	ASFE.C	U	1.50	439	186	119	285	76	38	201	1207	942	2149	1737.3
12	ASFE.C	U	1.90	409	184	133	270	71	38	191	1150	897	2047	1646.7
13	ASSEL.C	U	1.40	335	112	126	175	68	40	175	806	636	1442	1516.8
14	ASSEL.C	U	1.50	353	129	154	191	71	43	203	898	706	1604	1789.4
15	ASSEL.C	U	1.60	323	131	186	188	70	42	201	890	699	1589	1764.3
16	ASSEL.C	U	1.90	331	134	200	192	76	42	206	924	721	1645	1809.9
17	ATOI.C	U	1.40	11	7	3	8	5	19	10	39	33	72	113.9
18	BACKUP.C	U	1.40	14	13	12	7	2	9	10	30	22	52	61.7
19	BINREC.C	G	1.00	31	31	48	22	3	23	25	108	68	176	220.1
20	BLDWLIN.C	Q	1.00	96	24	34	51	22	32	28	267	154	421	294.6
21	BROSCIT.C	B	1.00	70	36	37	43	8	28	67	222	169	391	541.0
22	BTDTXT.C	U	1.40	269	99	119	158	54	32	173	624	480	1104	1446.2
23	BTDTXT.C	U	1.50	315	120	152	177	68	34	195	726	548	1274	1656.4
24	BTDTXT.C	U	1.80	321	126	165	179	71	35	198	739	554	1293	1690.1
25	BTDTXT.C	U	1.11	319	130	174	177	71	35	198	737	550	1287	1690.1
26	BTOCFE.C	U	1.40	104	49	84	54	19	30	97	269	202	471	787.4

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
1	2000.5	119713	2796.3	3326.7	2857.7	2514.1	185.6	233.3	368.8	154.1
2	302.9	3937	315.0	406.4	302.6	253.8	106.6	140.9	160.3	77.4
3	322.1	4569	340.7	436.6	327.0	274.8	106.6	140.9	165.6	77.4
4	1880.9	10191	2485.3	2908.5	2635.3	2326.7	226.5	282.2	285.9	187.4
5	2693.6	160878	3419.4	3986.7	3630.8	3172.9	367.4	444.2	448.1	308.3
6	2268.3	149711	3105.4	3639.7	3286.1	2918.0	226.5	282.2	304.6	187.4
7	2890.4	186602	3716.2	4347.8	3951.1	3462.6	367.4	444.2	468.4	308.3
8	1400.6	34314	1552.7	1851.6	1779.2	1377.3	378.1	456.3	494.7	312.9
9	435.0	5351	473.7	608.1	524.9	402.2	129.5	166.5	183.9	105.1
10	10441.3	751776	13308.0	15215.7	13699.8	12194.4	1158.1	1286.2	1602.4	1058.4
11	16979.0	1511888	21662.9	24565.3	22132.4	19921.8	1737.3	1897.7	2484.8	1619.6
12	16046.9	1431866	20491.2	23225.9	20930.8	18896.5	1646.7	1804.5	2381.0	1526.3
13	11172.9	812110	13704.6	15778.2	14084.8	12597.3	1516.8	1675.1	2052.9	1393.3
14	12739.8	952598	15491.1	17757.0	15973.5	14239.8	1789.4	1963.2	2369.3	1655.2
15	12592.5	919630	15324.9	17575.0	15799.3	14082.9	1764.3	1935.1	2341.6	1632.9
16	13084.7	961722	15948.1	18304.1	16420.6	14645.5	1809.9	1982.0	2453.4	1674.3
17	349.8	10965	372.6	451.9	344.3	290.2	113.9	140.9	152.4	74.6
18	220.9	2187	245.3	325.2	300.9	223.7	61.7	86.4	77.5	54.9
19	983.0	30747	1143.5	1357.5	1231.8	1021.2	220.1	275.1	313.9	182.2
20	2486.8	218839	271.3	3863.8	3334.0	2835.0	294.6	361.8	418.5	230.8
21	2568.8	90714	2981.1	3568.7	3218.1	2655.3	541.0	632.2	716.9	487.3
22	8478.1	376371	10069.4	11611.3	10688.5	9289.8	1446.2	1583.4	1770.0	1371.2
23	9987.1	477130	11885.5	13707.2	12597.7	10962.1	1656.4	1804.5	2022.4	1575.8
24	10168.4	497892	12091.3	13955.4	12801.6	11160.0	1690.1	1841.7	2067.3	1610.2
25	10121.2	492003	12027.1	13884.5	12707.2	11090.9	1690.1	1841.7	2082.5	1605.1
26	3291.7	102822	3718.2	4379.0	4078.0	3350.5	787.4	896.0	946.6	709.5

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
27	BTOCSEL.C	A	1.4	359	124	122	156	67	38	173	773	607	1380	1485.6
28	BTOCSEL.C	A	1.5	366	128	132	162	68	38	176	791	620	1411	1512.3
29	BTOCSEL.C	A	1.7	367	129	135	162	69	38	178	796	623	1419	1530.1
30	BUFF2OBJ.C	U	1.0	15	8	16	9	4	16	14	43	31	74	117.3
31	BUILDIDX.C	G	1.0	48	22	32	30	7	15	49	184	144	328	333.7
32	CDMENU.C	G	1.0	214	113	127	119	47	43	124	580	431	1011	1095.6
33	CDMENU.C	G	1.1	209	115	138	116	46	42	120	562	416	978	1055.3
34	CDMENU.C	G	1.2	210	116	138	117	46	42	120	564	418	982	1055.3
35	CDMENU.C	G	1.3	214	116	134	120	47	42	120	576	427	1003	1055.3
36	CHATTRW.C	Q	1.0	19	21	29	11	4	20	21	68	42	110	178.7
37	CHBORDW.C	Q	1.0	47	27	33	25	15	24	34	157	116	273	283.0
38	CHDIMW.C	Q	1.0	131	39	42	58	38	28	50	475	294	769	416.8
39	CHDISC.C	G	1.0	85	35	56	56	11	34	79	274	199	473	671.0
40	CHDISC.C	G	1.1	86	37	64	56	11	34	79	273	204	477	671.0
41	CHDISC.C	G	1.2	53	33	60	27	6	32	63	163	115	278	536.6
42	CHDISC.C	G	1.3	53	33	60	27	6	32	64	166	117	183	544.0
43	CHECKDS.C	G	1.4	63	35	38	32	10	23	55	159	120	279	422.0
44	CHECKLU.C	J	1.0	36	18	19	18	5	22	26	81	59	140	220.3
45	CHIMDISC.C	U	1.0	68	32	55	34	12	23	63	209	136	345	480.6
46	CHNGDSK.C	U	1.0	297	111	112	175	80	32	159	971	694	1665	1322.7
47	CHNGDSK.C	U	1.1	417	181	214	235	106	38	209	1259	916	2175	1810.3
48	CHNGDSK.C	U	1.4	401	184	227	237	91	37	207	1175	882	2057	1785.3
49	CHNGDSK.C	U	1.5	406	186	229	240	92	37	208	1184	888	2072	1794.4
50	CHNGNPNM.C	U	1.4	93	32	30	60	9	26	74	265	227	492	581.7
51	CHNGWIND.C		1.4	25	13	13	.	7	15	22	59	49	108	156.7
52	CHRPRT.C	U	1.4	13	12	10	6	3	15	11	29	21	50	96.7

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
27	10655.1	710321	13028.5	14977.0	13714.5	11972.5	1485.6	1638.3	1980.0	1378.8
28	16923.2	731110	13366.6	15347.2	14053.9	12285.7	1512.3	1665.9	2007.3	1405.2
29	11004.2	731778	13454.1	15442.8	14126.3	12362.4	1530.1	1684.3	2028.8	1418.8
30	363.1	6432	386.9	482.5	371.8	335.3	117.3	153.6	163.8	89.8
31	1968.0	43376	2416.8	2898.6	2751.7	2222.9	333.7	391.5	384.6	306.1
32	7464.9	57853	96.3	387.7	9247.4	8248.4	1095.6	1257.1	1666.1	992.8
33	7178.4	522586	8752.9	10010.4	8905.2	7945.8	1055.3	1213.1	1625.2	959.0
34	7207.7	527246	8794.4	10056.1	8948.6	7983.7	1055.3	1213.1	1616.7	959.0
35	7361.9	550116	9013.0	10307.6	9187.8	8183.1	1055.3	1213.1	1609.1	959.0
36	589.3	11787	640.4	787.2	717.7	563.9	178.7	226.5	222.5	140.7
37	1599.2	65474	1940.8	2459.0	2166.3	1703.1	283.0	347.1	407.1	233.6
38	4833.5	397892	6634.3	7571.1	6796.7	5907.9	416.8	498.0	630.7	355.8
39	3225.9	138244	3738.5	4323.6	3865.1	3335.9	671.0	803.9	1007.3	579.4
40	3253.2	142813	3774.5	4370.0	3904.9	3375.5	671.0	799.2	1013.9	582.1
41	1826.4	53343	1985.1	2459.0	2179.8	1786.6	536.6	632.2	705.0	466.2
42	1863.5	54509	2028.1	2507.4	2213.9	1824.8	544.0	640.2	723.2	473.5
43	1753.0	44000	1991.6	2410.7	2139.7	1747.7	422.0	498.0	551.7	372.5
44	781.9	19517	860.6	1058.4	925.0	750.0	220.3	275.1	284.6	183.4
45	2217.1	55039	2574.7	3076.9	2751.9	2263.1	480.6	560.5	578.5	422.6
46	12616.4	881086	16186.1	18621.0	16579.0	14922.4	1322.7	1456.3	1647.2	1189.3
47	17287.7	1439594	21978.8	25302.5	22714.8	20281.4	1870.3	1987.2	2325.7	1669.2
48	16313.5	1285932	20613.2	23884.0	21381.2	19076.5	1785.3	1959.1	2208.0	1645.1
49	16444.7	1298816	20785.4	24084.6	21564.5	19242.0	1794.4	1968.5	2225.1	1654.1
50	3268.8	130354	3909.8	4232.1	4110.0	3529.4	581.7	641.2	728.7	520.9
51	562.6	9398	622.0	770.7	672.2	516.4	156.7	199.4	225.1	125.2
52	235.0	3365	233.1	303.6	255.6	185.9	96.7	128.4	135.3	67.3

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
53	CLEANUP.C	U	1.4	63	10	6	30	13	29	51	141	132	273.0	430.3
54	CLMDOC.C	G	1.0	18	15	20	9	4	16	24	60	43	103.0	174.0
55	CLOSES.C	Q	1.0	26	25	32	16	4	19	28	86	59	145.0	215.3
56	CLOSESF.C	Q	1.0	34	26	33	19	8	22	29	107	70	177.0	239.0
57	CLOSEV.C		1.4	29	17	21	.	4	21	36	78	79	157.0	278.4
58	CLOSEW.C	Q	1.0	33	24	31	22	7	20	25	101	68	169.0	202.5
59	CLRALL.C	U	1.0	19	8	20	12	4	18	19	61	45	106.0	155.8
60	CLRMARKS.C	K	1.0	36	13	23	16	10	24	34	108	82	190.0	283.0
61	CLROBJ.C	U	1.0	43	12	19	23	16	22	39	156	111	267.0	304.2
62	CONJEXPR.C	H	1.4	40	11	22	17	14	22	31	111	70	181.0	251.7
63	CONSTAT.C	U	1.4	4	2	3	1	1	2	3	4	3	7.0	6.8
64	COPYMDL.C	G	1.0	28	19	22	13	5	20	33	92	70	162.0	252.9
65	CREATEV.C		1.4	45	22	25	.	4	17	46	128	116	244.0	323.6
66	CTRLFPOS.C		1.4	68	19	18	.	20	25	36	166	120	286.0	302.2
67	CURPOS.C	D	1.4	12	10	10	8	1	12	17	31	34	65.0	112.5
68	CURSOR.C	B	1.4	31	15	13	13	4	15	25	54	61	115.0	174.7
69	DDOCNUM.C	G	1.0	82	20	45	64	11	18	25	213	193	406.0	191.2
70	DISKOK.C	U	1.4	12	7	13	6	5	8	9	28	20	48.0	52.5
71	DISPDET.C	B	1.0	111	45	48	73	10	31	118	358	292	650.0	965.7
72	DISPFILE.C		1.4	34	22	22	.	5	22	31	90	64	154.0	251.7
73	DISPINIT.C		1.4	44	26	28	.	7	20	44	165	128	293.0	326.7
74	DISPINIT.C		1.6	49	31	37	.	9	23	50	198	146	344.0	386.2
75	DISPLN.C		1.4	13	14	11	.	2	15	18	36	29	65.0	133.7
76	DISPMSG.C	U	1.0	30	7	20	18	6	18	23	74	49	123.0	179.1
77	DISPSCR.C	U	1.0	161	16	30	75	41	30	58	328	235	563.0	487.0
78	DISPTARG.C	A	1.4	31	24	40	24	1	17	30	71	59	130.0	216.7

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
53	1725.9	64772	1936.5	2363.9	2162.3	1715.3	430.2	502.2	574.3	369.6
54	548.2	7857	587.7	676.8	629.7	486.4	174.0	221.6	234.2	135.7
55	805.4	16123	899.7	1145.3	1067.6	802.7	215.3	268.1	269.5	174.9
56	1004.0	26658	1150.4	1438.3	1233.4	1014.2	239.0	296.4	329.1	193.5
57	915.8	21101	988.3	1268.7	1132.9	854.4	278.4	339.3	385.6	238.3
58	928.1	25245	1086.4	1321.8	1189.0	973.9	202.5	254.1	279.2	165.7
59	552.2	11771	608.9	745.9	584.2	537.7	155.8	199.4	242.5	124.0
60	117.0	3221	1250.8	1547.0	1245.4	1063.7	283.0	347.1	436.7	234.9
61	1583.5	49576	1890.7	2228.4	1888.8	1687.3	304.2	369.2	407.5	260.2
62	1036.8	25752	1183.2	1429.3	1202.1	1016.5	251.7	310.8	360.9	207.7
63	16.3	16	12.8	24.0	24.0	12.8	6.8	11.6	11.6	6.8
64	927.9	16983	1029.2	1107.8	1056.5	923.5	252.9	331.6	341.2	211.6
65	1458.5	31262	1691.5	2109.9	2097.2	1454.6	323.6	397.7	388.8	287.9
66	1696.2	70675	2053.1	2449.3	2080.0	1757.8	302.2	369.2	452.7	247.4
67	315.8	3789	326.6	467.2	467.2	276.6	112.5	147.2	147.2	92.0
68	612.0	11200	672.5	1015.3	887.5	507.8	174.7	219.7	254.6	136.3
69	2203.1	153069	3112.8	3690.5	3105.0	2799.2	191.2	240.2	354.9	157.7
70	196.2	1744	221.0	275.1	227.9	182.7	52.5	69.5	72.1	37.1
71	4692.5	179984	5428.6	6355.0	5924.0	4902.3	965.7	1084.3	1193.5	900.7
72	882.1	20032	968.3	1091.9	1001.1	351.4	251.7	316.2	351.4	205.7
73	1758.0	51142	2111.4	2478.3	2085.1	1816.2	326.7	391.5	458.7	277.1
74	2129.3	71502	2560.3	2987.6	2515.5	2214.3	386.2	459.5	540.0	330.7
75	327.9	3962	327.0	413.9	381.4	271.7	133.7	173.0	169.1	104.0
76	659.0	12635	734.6	912.9	717.6	654.9	179.1	226.5	261.4	153.0
77	3636.7	221021	4592.3	5484.0	4736.9	4172.4	487.4	576.3	650.0	429.0
78	722.1	12071	783.7	938.4	938.4	699.7	216.7	268.1	268.1	184.7

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
79	DISPTOS.C		1.4	22	24	46	.	5	24	22	82	52	134.0	208.1
80	DISPTOS.C		1.5	22	24	46	.	5	24	22	82	52	134.0	208.1
81	DISPWIND.C		1.4	37	28	34	.	7	26	33	144	114	258.0	288.7
82	DLCLOSE.C	A	1.4	50	16	38	19	12	21	34	124	79	203.0	265.2
83	DLCOSE.C	G	1.5	10	16	25	6	1	13	13	21	17	38.0	96.2
84	DLCREATE.C	O	1.4	49	21	30	27	6	24	40	122	81	203.0	322.9
85	DLCREATE.C	O	1.5	36	27	30	20	3	24	51	129	96	225.0	399.3
86	DLDELETE.C	A	1.4	23	16	27	9	3	17	23	48	36	84.0	173.5
87	DLDELETE.C	G	1.5	9	13	17	4	1	12	12	23	17	40.0	86.0
88	DLEXTEND.C	A	1.4	22	16	31	14	2	14	3	8	1	8.4	.
89	DLFREE.C	O	1.4	17	12	27	12	2	14	14	40	29	69.0	106.6
90	DLFTOI.C	A	1.4	22	16	32	16	4	22	28	77	52	129.0	232.7
91	DLFTOI.C	A	1.5	48	19	34	30	8	26	52	144	98	242.0	418.6
92	DLFTOI.C	G	1.6	37	24	36	17	4	22	46	138	94	232.0	352.2
93	DLINIT.C	U	1.4	41	19	34	30	6	22	37	117	84	201.0	290.9
94	DLINIT.C	U	1.5	19	16	28	11	3	16	21	61	39	100.0	156.2
95	DLITOF.C	O	1.4	81	28	46	35	10	26	59	178	129	307.0	469.3
96	DLITOF.C	O	1.6	12	12	19	6	1	12	15	36	28	64.0	101.6
97	DLOPEN.C	A	1.4	35	17	32	18	7	23	38	106	74	180.0	303.5
98	DLOPEN.C	U	1.5	29	25	29	17	2	23	45	104	76	180.0	351.2
99	DLREAD.C	A	1.4	74	23	48	40	12	28	54	205	146	351.0	445.4
100	DLREAD.C	A	1.5	54	27	29	29	6	23	48	157	106	263.0	372.1
101	DLREINIT.C	O	1.4	22	16	28	20	4	16	22	73	49	122.0	162.1
102	DLSHORT.C	M	1.0	231	77	93	138	46	28	94	614	467	1081.0	750.7
103	DLSHORT.C	M	1.2	200	85	106	115	37	29	85	515	386	901.0	685.7
104	DLSHORT.C	M	1.3	201	87	111	115	37	30	88	520	389	909.0	715.6

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
79	740.2	20994	817.7	972.4	789.7	705.1	208.1	261.1	320.8	165.0
80	740.2	20994	817.7	972.4	789.7	705.1	208.1	261.1	320.8	165.0
81	1517.7	68160	1811.4	2123.7	1834.7	1579.1	288.7	354.4	464.5	241.3
82	1173.6	28633	1360.3	1566.4	1358.2	1194.9	265.2	326.3	343.4	218.5
83	178.6	1518	161.7	212.9	212.9	132.6	96.2	128.4	128.4	72.7
84	1218.0	29597	1359.1	1629.2	1441.6	1204.9	324.9	319.5	412.4	374.1
85	1401.5	31657	1536.6	1860.3	1680.8	1303.5	399.3	474.8	531.2	336.2
86	447.0	5948	454.2	584.3	528.1	406.4	173.5	219.7	214.9	144.1
87	183.4	1559	173.5	240.2	240.2	152.7	86.0	116.1	116.1	73.3
88	430.0	5853	468.8	576.3	548.6	402.1	128.4	166.5	161.6	99.5
89	331.7	4810	353.8	436.6	380.3	312.1	106.6	140.9	155.4	85.3
90	728.1	14873	779.0	880.2	786.4	691.8	232.7	292.1	315.6	192.2
91	418.6	37266	1680.7	1950.0	1673.4	1498.3	418.6	498.8	597.0	358.7
92	1412.3	31746	1597.1	1712.8	1612.5	1414.0	352.2	465.8	493.2	297.1
93	1182.4	29528	1340.8	1583.4	1394.9	1157.7	290.9	354.4	359.5	249.2
94	520.9	7740	567.9	705.0	660.7	488.5	156.2	199.4	188.5	125.9
95	1967.7	55929	2235.1	2553.4	2387.1	2025.7	469.3	552.5	573.3	410.5
96	304.3	3408	320.7	421.5	421.5	292.2	101.6	134.6	134.6	88.5
97	1067.5	23907	1172.7	1420.3	1214.0	1026.4	303.5	369.2	384.2	253.0
98	1095.7	21282	1171.7	1438.3	1326.2	986.2	351.2	421.5	440.9	295.2
99	2231.5	84466	2624.0	2972.5	2717.4	2339.6	445.4	539.0	570.3	385.0
100	1617.4	41075	1858.4	2247.5	2123.1	1605.8	372.1	444.2	441.9	324.8
101	640.2	11408	727.0	870.7	762.5	605.7	162.1	206.1	212.7	130.6
102	792.1	521101	9827.9	11576.5	9995.1	8933.4	750.7	853.9	1181.1	684.6
103	6156.4	405383	7956.0	9385.9	8179.1	7185.9	685.7	787.2	1037.9	622.3
104	6526.3	414837	8038.4	9487.9	8272.7	7267.1	715.6	820.5	1075.6	651.3

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
105	DLWRITE.C	A	1.40	99	32	49	38	13	28	56	191	137	328	459.8
106	DLWRITE.C	A	1.50	99	32	49	38	13	28	56	191	137	328	459.8
107	DLWRITE.C	U	1.60	22	21	23	10	3	21	26	73	50	123	214.5
108	DNSCROLL.C		1.40	45	38	23	.	10	23	36	133	105	138	190.2
109	DNSCROLL.C		1.60	41	20	27	.	10	23	33	117	89	206	270.5
110	DOCINFO.C	E	1.40	31	24	38	24	5	22	26	108	65	173	220.3
111	DOCSTAMP.C		1.40	26	13	18	.	4	24	27	72	53	125	238.4
112	DOCTHERE.C	G	1.00	30	23	32	8	11	16	15	68	51	119	122.6
113	DOWNOBJ.C	Q	1.00	23	21	28	13	7	19	19	82	61	143	161.4
114	DSETSCR.C		1.40	77	49	41	.	14	24	42	239	195	434	336.5
115	DSKRESET.C	U	1.00	8	2	0	4	1	10	11	16	16	32	71.3
116	DUMPDTOV.C		1.40	22	9	11	.	6	18	19	46	36	82	155.8
117	DWNLDLNG.C	N	1.00	192	74	95	115	36	31	133	574	409	983	1091.9
118	DWNLDLNG.C	N	1.20	203	76	99	123	37	31	133	605	436	1041	1091.9
119	DWNLDLNG.C	N	1.30	205	78	104	123	38	32	136	612	441	1053	1123.9
120	EDERROR.C	U	1.00	63	18	24	14	5	23	82	213	164	377	625.4
121	EDGETKE.C	U	1.00	15	7	18	7	3	14	16	38	26	64	117.3
122	EDITOBJ.C	U	1.00	93	12	21	45	20	28	54	328	251	571	445.4
123	EDITOBJ.C	U	1.00	136	29	3	79	32	38	50	366	254	620	481.6
124	EDITSCR.C	U	1.00	207	22	36	107	47	32	70	445	320	765	589.0
125	EDITST.C	B	1.40	373	212	234	193	76	37	133	897	724	1621	1131.1
126	EDOBJ.C	U	1.00	126	24	23	77	28	37	49	356	266	622	.
127	EDOBJ2.C	U	1.00	119	17	24	63	47	34	41	353	243	596	392.6
128	EDOBJ3.C	U	1.00	101	23	23	63	24	37	44	282	199	481	433.0
129	ENDOBJ.C	Q	1.00	18	20	27	10	5	18	16	60	47	107	139.1
130	ERRPAUSE.C	K	1.40	78	31	32	45	17	26	57	216	146	362	454.7

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
105	2096.7	71811	2419.7	2775.9	2567.1	2180.4	459.8	552.5	600.7	396.3
106	2096.7	71811	2419.7	2775.9	2567.1	2180.4	459.8	552.5	600.7	396.3
107	683.2	13796	734.1	921.4	845.7	617.0	214.5	268.1	262.7	172.1
108	1400.1	46961	1643.3	1963.2	1643.6	1439.1	290.2	354.4	456.6	246.6
109	1196.3	37104	1380.2	1665.9	1397.9	1192.2	270.5	332.5	400.3	227.2
110	966.2	26571	1121.0	1339.6	1162.7	973.3	220.3	275.1	326.4	182.6
111	709.1	16702	747.8	929.9	846.4	648.8	238.4	296.4	300.4	198.1
112	589.5	16036	703.2	870.7	689.4	603.4	122.6	160.0	220.8	93.3
113	350.5	22889	883.1	1119.1	971.9	772.6	161.4	206.1	239.0	124.5
114	2623.3	146154	3371.7	3915.0	3563.4	2980.4	336.5	406.4	466.3	293.1
115	140.6	1022	128.0	186.1	186.1	108.7	71.3	98.1	98.1	54.9
116	427.2	7285	440.2	560.5	437.2	358.1	155.8	199.4	215.5	119.6
117	7232.5	344739	8809.1	10101.6	8967.3	7956.5	1091.9	1272.9	1528.1	1015.0
118	7659.2	389180	9413.6	10732.6	9586.6	8518.4	1091.9	1281.7	1545.1	1015.0
119	7784.1	403858	9539.5	10882.0	9721.5	8638.9	1123.9	1317.0	1584.3	1046.1
120	2531.3	58219	2854.1	3296.6	3015.8	2699.6	625.4	713.2	659.7	555.8
121	314.0	3572	321.6	413.9	332.3	291.3	117.3	153.6	159.7	102.0
122	3681.0	239539	4742.1	5452.0	4990.0	4342.4	445.4	529.1	594.5	400.9
123	4004.8	386548	5145.9	5890.7	5249.3	4670.1	481.6	576.3	679.0	409.8
124	5104.4	373351	6578.0	7770.5	6859.8	5962.4	589.0	688.7	849.0	528.9
125	12010.6	1209551	15676.5	18036.4	16353.5	14405.0	1131.1	1268.5	1763.0	1055.0
126	3997.1	401427	5160.1	5933.8	5174.2	4678.4	467.9	560.5	757.0	397.3
127	3712.4	374045	4913.4	5644.1	4703.5	4348.1	392.6	474.8	639.5	330.8
128	3049.5	255150	3815.0	4399.7	3796.3	3441.9	433.0	521.3	685.5	366.6
129	544.5	14392	615.5	795.5	699.9	543.8	139.1	179.5	197.9	103.7
130	2307.8	76845	2724.8	3256.5	2749.0	2434.9	454.7	537.0	593.4	395.9

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
131	EXNPWRIT.C	G	1.40	82	42	50	35	12	25	80	281	209	490	621.9
132	EXNPWRIT.C	G	1.50	83	42	50	35	12	27	82	287	211	498	649.7
133	EXPR.C	H	1.40	42	12	18	16	12	14	22	84	61	145	151.4
134	FIELDED.C	U	1.40	199	69	67	130	42	34	93	496	355	851	781.1
135	FIELDED.C	U	1.50	196	70	72	130	41	34	90	496	352	858	757.2
136	FILLBUFF.C	U	1.00	233	51	56	128	45	39	96	782	553	1335	838.3
137	FINDDOC.C	U	1.40	155	34	8	91	21	24	43	371	263	634	343.4
138	FINDIM.C	G	1.00	45	22	22	28	7	26	39	147	103	250	328.3
139	FINDIM.C	G	1.10	45	25	32	28	7	25	39	144	108	252	322.2
140	FINDIM.C	G	1.20	47	37	43	27	8	26	43	145	114	259	355.5
141	FINDOP.C	B	1.40	44	18	50	19	3	18	31	83	64	147	228.6
142	FINDOP.C	B	1.60	51	23	70	22	3	24	40	106	75	181	322.9
143	FINDUNIT.C		1.40	36	13	16	.	9	25	27	93	64	157	244.5
144	FIREUP.C	U	1.40	24	25	38	13	2	19	35	77	48	125	260.2
145	FIREUP.C	U	1.60	25	27	41	14	2	19	36	80	50	130	266.8
146	FIREUP.C	U	1.80	27	27	43	14	2	19	38	88	55	143	280.1
147	FIREUP.C	U	1.90	31	31	48	16	3	21	22	99	61	160	318.7
148	FIREUP.C	U	1.11	40	36	56	20	6	25	54	134	83	217	426.9
149	FLDTOBUF.C		1.40	35	19	21	.	7	22	26	93	65	158	220.3
150	FLIPATTR.C		1.40	20	18	31	.	6	22	22	72	48	120	196.2
151	FLIPMANY.C		1.40	30	23	31	.	8	29	28	106	73	179	275.5
152	FLOPPIES.C	U	1.40	154	93	139	84	24	31	133	473	395	868	1091.9
153	FREEBUF.C	U	1.00	10	5	13	7	2	14	8	38	19	57	77.3
154	FREEDS.C	G	1.40	16	17	21	10	2	18	23	50	45	95	179.1
155	FREESCR.C	Q	1.00	21	21	26	10	3	14	12	52	26	78	96.3
156	GENBORD.C	Q	1.00	145	37	56	54	15	30	69	428	304	732	568.7

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
131	3290.0	107438	3896.6	4186.4	4064.2	3494.5	621.9	739.4	813.4	569.1
132	3370.6	117086	3972.5	4276.7	4150.4	3572.7	649.7	771.3	842.3	595.5
133	749.6	14550	898.7	1136.5	980.6	780.5	151.4	192.7	216.7	122.4
134	5947.4	385940	7448.7	8618.8	7632.2	6793.0	781.1	896.0	1148.3	707.4
135	5897.2	392096	7419.0	8585.2	7600.0	6763.7	757.2	870.7	1123.8	683.8
136	9447.5	1061225	12544.3	14441.5	12588.3	11390.8	838.3	963.9	1441.3	735.4
137	3845.9	282271	5280.8	6041.4	5508.5	4785.4	343.4	430.4	502.8	294.4
138	1505.6	59692	1747.1	2003.1	1799.4	1496.3	328.3	409.8	485.5	270.3
139	1512.0	52339	1762.0	1973.2	1798.8	1511.2	322.2	425.8	496.0	265.0
140	1582.1	54528	1820.0	2095.5	1956.9	1600.4	355.5	445.5	462.6	302.0
141	825.4	15336	913.1	1119.1	1051.4	799.4	228.6	282.2	287.3	193.7
142	1086.0	24435	1180.3	1447.3	1375.1	1029.9	322.9	391.5	395.0	277.9
143	895.0	26518	992.1	1206.6	1004.7	846.5	244.5	303.6	365.8	198.5
144	719.4	9372	750.6	921.4	921.4	636.1	260.2	318.0	318.0	219.2
145	751.6	9917	787.9	963.9	963.9	666.9	266.8	325.2	325.2	225.6
146	834.1	11469	886.4	1093.0	1093.0	753.1	280.1	339.8	339.8	230.6
147	956.4	14585	1018.1	1250.7	1229.0	873.8	318.7	384.0	378.6	275.5
148	1367.9	26282	1476.0	1785.9	1669.2	1291.7	426.9	505.8	510.5	374.5
149	882.4	24267	999.6	1215.4	1015.3	842.5	220.3	275.1	322.0	172.3
150	655.1	15723	712.3	853.9	764.3	611.1	196.2	247.1	250.1	153.2
151	1044.1	38470	1165.0	1393.3	1130.9	1010.0	275.5	339.8	421.8	221.6
152	6386.4	293988	7610.1	8572.5	8138.0	6976.8	1091.9	1312.9	1405.9	994.2
153	254.2	4226	280.1	347.1	293.6	226.5	77.3	104.0	118.8	53.7
154	509.0	8962	529.3	737.7	717.9	460.4	179.1	226.5	221.7	143.7
155	366.6	5561	418.6	529.1	470.4	358.6	96.3	128.4	135.7	75.3
156	4852.7	320700	6248.7	7262.1	6482.7	5608.0	568.7	664.4	749.5	499.7

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
157	GENINDEX.C	U	1.4	41	16	39	19	6	22	30	110	78	188	245.3
158	GENSWICH.C	G	1.0	60	35	42	32	10	21	43	148	106	254	325.6
159	GENSWICH.C	G	1.1	23	22	42	13	3	16	20	61	40	101	150.4
160	GENSWICH.C	G	1.2	23	22	42	13	3	16	21	63	40	103	156.2
161	GENW.C	Q	1.0	55	26	41	44	7	29	36	222	126	348	327.0
162	GENWBDD.C	Q	1.0	39	24	38	22	9	22	28	131	72	203	232.7
163	GETBITMA.C	U	1.0	24	8	0	8	4	16	25	67	43	110	180.1
164	GETBITS.C	U	1.4	13	9	9	8	2	15	12	49	28	77	101.6
165	GETCANDA.C		1.4	12	11	20	.	1	16	15	49	30	79	122.6
166	GETCAROU.C	I	1.0	8	13	20	5	4	13	15	30	24	54	106.7
167	GETCFDB.C		1.4	43	18	3	.	5	21	45	138	103	241	339.4
168	GETCFDB.C		1.6	43	19	6	.	5	21	45	138	103	241	339.4
169	GETCH.C	H	1.4	120	33	37	67	35	30	63	373	220	593	523.8
170	GETCLASS.C		1.4	47	34	39	.	11	27	37	137	94	231	321.1
171	GETCONFG.C		1.4	24	22	34	.	3	17	28	62	46	108	204.1
172	GETCTRLF.C		1.4	28	28	44	.	4	17	28	68	52	120	204.1
173	GETCTRLF.C		1.6	41	40	67	.	6	22	41	104	76	180	317.8
174	GETCTRLF.C		1.8	42	45	80	.	7	18	40	105	76	181	287.9
175	GETDATE.C	U	1.4	12	4	0	8	1	11	17	38	33	71	107.5
176	GETDEV.C		1.4	59	23	16	.	8	22	62	166	101	267	467.3
177	GETDL.C	A	1.4	153	32	49	61	13	34	62	283	211	494	542.1
178	GETDL.C	A	1.6	127	43	119	69	14	37	79	333	242	575	690.7
179	GETDLEDT.C		1.0	55	23	27	.	10	22	35	167	106	273	277.6
180	GETDLOPT.C		1.0	79	26	26	.	8	25	40	214	137	351	329.0
181	GETDLOPT.C		1.1	81	28	30	.	8	26	41	219	140	359	341.9
182	GETDLOPT.C		1.2	94	34	38	.	10	28	47	255	260	415	395.7

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
157	1071.7	30650	1236.2	1492.5	1325.9	1077.1	245.3	303.6	346.3	208.0
158	1524.0	39447	1780.2	2104.8	1989.0	1599.4	325.6	391.5	415.1	276.9
159	522.2	8355	574.7	713.2	643.5	493.8	150.4	192.7	206.2	121.9
160	536.6	8176	589.4	729.5	657.1	501.4	156.2	199.4	222.5	124.6
161	2095.8	106361	2609.5	3017.4	2744.8	2368.0	327.0	398.9	438.2	273.9
162	1145.7	32407	1365.6	1610.0	1425.1	1174.1	232.7	289.3	291.7	187.4
163	589.3	8109	639.8	748.3	696.4	533.4	190.1	241.4	260.2	149.9
164	366.1	6407	409.7	490.3	447.2	343.3	101.6	128.4	128.0	71.5
165	391.4	6262	422.3	513.5	513.5	359.6	122.6	160.0	160.0	95.4
166	259.6	2700	257.2	332.5	244.9	212.1	106.7	140.9	151.7	80.9
167	1456.7	35009	1669.7	1737.8	1749.7	1489.3	339.4	465.8	461.9	287.7
168	1456.7	35009	1669.7	1711.8	1749.0	1489.3	339.4	465.8	461.9	287.7
169	3877.7	203119	4898.5	5729.8	4927.6	4311.8	523.8	616.1	870.6	450.9
170	1386.0	47536	1588.6	1916.4	1683.2	1371.6	321.1	391.5	437.1	271.1
171	593.1	8283	623.2	770.7	694.6	539.2	204.1	254.1	254.2	167.8
172	659.0	10403	710.4	879.1	782.8	632.0	204.1	254.1	243.5	168.4
173	1075.9	21938	1171.7	1447.3	1307.8	1048.4	317.8	384.0	380.3	273.5
174	1060.3	18131	1179.8	1447.3	1311.8	1069.0	287.9	347.1	337.6	250.0
175	341.3	3644	365.9	513.7	513.5	327.8	107.5	140.9	140.9	94.5
176	1706.7	30584	1896.7	2314.5	1971.5	1750.7	467.3	554.8	605.8	417.6
177	3253.0	188200	3934.1	4597.6	4176.7	3495.4	542.1	640.2	770.1	467.3
178	3943.3	223473	4706.7	5505.3	4993.8	4187.6	690.7	803.8	966.6	601.6
179	1592.4	53049	1946.2	2324.1	1918.9	1761.6	277.6	339.8	368.6	239.8
180	2113.9	90499	2629.1	3196.5	2644.5	2362.1	329.0	398.9	477.4	279.6
181	2177.7	96670	2700.8	3276.6	2714.6	2425.9	341.9	413.9	491.1	291.8
182	2585.0	123198	3210.1	3874.0	3239.1	2895.2	395.7	474.8	574.2	341.6

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
183	GETDOCMX.C		1.0	25	21	38	.	6	16	27	92	55	147	192.4
184	GETFIELD.C	G	1.0	12	13	21	7	1	13	12	35	24	59	91.1
185	GETFREE.C		1.4	89	55	82	.	15	31	52	293	225	518	450.0
186	GETIDNUM.C	G	1.0	12	17	23	7	1	13	14	33	24	57	101.4
187	GETIMGPT.C	J	1.0	14	16	18	5	3	15	17	42	24	66	128.1
188	GETIMIDX.C	G	1.0	13	13	23	7	2	16	14	35	21	56	117.3
189	GETITLE.C		1.4	31	23	35	.	6	24	27	99	64	163	238.4
190	GETITLE.C		1.5	35	24	37	.	7	25	31	113	72	185	269.7
191	GETKEY.C	L	1.4	128	39	51	31	18	30	204	456	385	841	1712.4
192	GETLDRBN.C	D	1.4	249	157	173	154	53	39	188	768	577	1345	1626.4
193	GETLDRBN.C	D	1.6	256	157	176	157	55	40	191	806	600	1406	1660.2
194	GETLDRBN.C	D	1.7	273	160	184	172	56	40	194	851	642	1493	1687.3
195	GETLFMT.C		1.4	24	12	12	.	5	16	22	71	40	111	162.1
196	GETMON.C	U	1.4	10	14	17	4	2	13	17	29	25	54	117.6
197	GETNBLK.C	U	1.4	15	13	20	6	2	15	18	42	27	69	133.7
198	GETNPFN.C		1.4	32	23	29	.	5	20	31	98	66	164	240.0
199	GETOBJI.C	Q	1.0	22	20	34	14	5	20	23	82	59	141	190.5
200	GETOBJL.C	Q	1.0	13	16	25	6	4	14	16	40	29	69	117.3
201	GETOBJN.C	Q	1.0	13	16	25	6	4	14	14	38	25	63	106.6
202	GETOBJS.C	Q	1.0	48	21	33	22	15	29	36	150	79	229	263.1
203	GETOPTIO.C		1.4	45	14	15	.	8	21	28	123	78	201	226.8
204	GETOPTIO.C		1.6	73	21	20	.	14	21	33	201	132	333	258.7
205	GETOPTIO.C		1.7	87	25	22	.	17	21	25	240	158	398	271.8
206	GETORDER.C		1.4	33	27	41	.	8	23	30	132	94	226	241.2
207	GETORDER.C		1.6	40	30	49	.	11	25	37	170	112	282	308.8
208	GETPAGE.C		1.4	27	15	26	.	6	21	23	80	70	150	196.3

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
183	797.7	12999	918.1	1084.3	938.1	798.8	192.4	240.2	229.5	154.8
184	274.0	3562	289.6	369.2	369.2	243.6	91.1	122.2	122.2	68.8
185	3302.3	221474	4159.2	4796.4	4197.3	3680.3	450.0	537.0	731.3	372.5
186	271.0	3020	276.5	354.4	354.4	239.0	101.4	134.6	134.6	78.9
187	330.0	3494	336.5	436.6	373.2	287.6	128.1	166.5	170.3	106.1
188	274.8	3297	271.8	347.1	289.1	224.6	117.3	153.6	167.3	87.7
189	924.6	26300	1040.3	1268.5	1078.7	906.9	238.4	296.4	359.0	202.2
190	1074.4	31191	1214.9	1465.3	1259.7	1068.0	269.7	332.5	400.0	228.4
191	6619.0	187315	7334.5	8394.6	7783.7	6650.3	1712.4	1851.0	1899.1	1469.7
192	10526.7	630007	12653.7	14178.7	13376.2	11438.9	1626.4	1865.0	2151.7	1500.4
193	11039.6	693585	13318.9	14916.2	14070.3	12056.8	1660.2	1889.7	2181.2	1533.2
194	11750.5	777711	14270.0	15966.0	15050.6	12887.2	1687.3	1917.5	2208.9	1556.4
195	582.5	8473	649.7	812.2	706.3	545.4	162.1	206.1	221.6	130.3
196	265.0	2533	257.0	347.1	347.1	227.7	117.6	153.6	153.6	102.0
197	348.1	3916	354.9	459.5	441.4	292.2	133.7	173.0	168.5	106.0
198	930.3	19806	1047.2	1087.6	1065.1	922.0	240.8	304.2	337.2	200.5
199	765.1	19627	868.4	1058.4	959.4	743.6	190.5	240.2	244.7	145.9
200	338.6	4296	353.8	467.2	409.2	306.6	117.3	153.6	147.1	90.8
201	302.9	3786	315.5	406.4	351.7	269.7	106.6	140.9	135.3	80.8
202	1323.9	58329	1582.3	1916.4	1554.4	1342.9	263.1	325.2	389.6	203.5
203	1128.6	33010	1344.2	1629.2	1312.2	1183.2	226.8	282.2	290.3	190.3
204	1916.4	804877	2467.7	2928.3	2388.4	2213.2	258.7	318.0	321.1	221.6
205	2311.3	109557	3051.7	3599.1	2951.9	2753.3	271.8	332.5	333.7	234.5
206	1294.5	46646	1546.0	1860.3	1575.3	1353.6	251.2	310.8	372.0	199.0
207	1679.1	63533	2022.0	2410.7	2042.2	1773.4	308.8	376.6	462.8	249.5
208	818.9	26170	934.8	1119.2	1040.3	797.1	196.3	281.4	359.4	153.8

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
209	GETPBLK.C	U	1.4	14	16	20	5	2	16	19	43	29	72	144.7
210	GETPROPT.C		1.4	37	13	10	.	8	19	28	118	71	189	215.3
211	GETREFFI.C	A	1.4	24	18	31	.	5	20	36	90	67	157	272.6
212	GETSCRN.C		1.4	19	16	18	.	6	14	16	52	36	88	117.3
213	GETSEG.C		1.4	22	11	13	.	5	20	22	75	51	126	184.5
214	GETSEG.C		1.6	25	11	16	.	6	21	28	95	60	155	226.8
215	GETSHREC.C		1.0	124	80	113	.	24	28	69	401	268	669	556.1
216	GETSTAMP.C	U	1.4	37	17	18	36	7	27	63	236	162	398	505.0
217	GETTAB.C		1.4	23	17	25	.	4	18	20	73	45	118	161.5
218	GETTAB.C		1.5	23	17	25	.	4	18	20	73	45	118	161.5
219	GETTAGS.C		1.4	20	12	12	.	4	20	21	62	36	98	178.7
220	GETTIME.C	U	1.4	12	4	0	8	1	11	16	40	33	73	102.1
221	GETTOKEN.C	H	1.4	171	44	68	71	40	30	127	851	744	1595	1034.8
222	GETUNIT.C		1.4	51	11	14	.	27	24	28	142	102	244	244.6
223	GETUNIT.C		1.6	68	11	17	.	32	25	30	183	131	314	263.3
224	GETVOL.C	U	1.0	16	5	0	8	2	17	18	46	28	74	144.5
225	GETVOL.C	U	1.1	21	12	12	11	3	18	22	57	36	93	173.2
226	GETVOL.V	U	1.2	22	13	17	12	3	18	21	59	37	96	167.3
227	GETWAIT.C		1.4	23	12	18	.	4	16	23	84	46	130	168.0
228	GFINDDEC.C	U	1.4	33	14	29	14	4	18	19	77	53	130	155.8
229	GFINDUN.C	U	1.4	28	14	20	14	4	18	19	77	54	131	155.8
230	GTDCCNT.C	K	1.4	38	17	21	20	5	24	27	91	66	157	238.4
231	GTDOCTXT.C	U	1.4	58	36	59	30	11	27	70	202	141	343	557.4
232	GTDOCTXT.C	U	1.6	63	40	68	32	13	20	67	209	143	352	541.0
233	GTDOCTXT.C	U	1.7	69	43	72	37	14	28	67	230	160	390	541.0
234	GTHDRTIT.C	B	1.4	71	45	60	44	16	33	45	217	142	359	413.6

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
209	369.3	4510	374.2	482.5	445.5	305.1	144.7	186.1	179.7	112.8
210	1049.8	25289	1248.0	1537.9	1232.6	1095.2	215.3	261.0	283.9	184.2
211	911.8	16969	990.7	1147.6	1081.4	886.0	272.6	340.7	339.4	222.2
212	431.8	6801	482.5	592.2	507.1	407.6	117.3	153.6	174.0	90.1
213	679.6	1576	756.5	921.4	763.8	649.4	184.5	233.3	262.8	144.5
214	870.3	19581	978.5	1189.1	982.2	841.4	226.8	282.2	333.9	183.5
215	4415.3	240092	5629.3	6626.7	5830.9	5044.7	556.1	608.2	848.1	476.7
216	2583.8	89693	3049.4	3467.6	3157.7	2682.8	505.0	584.3	608.5	408.4
217	619.3	12540	699.0	870.7	731.7	597.2	161.5	206.1	232.2	131.5
218	619.3	12540	699.0	870.7	731.7	597.2	161.5	206.1	232.2	131.5
219	525.0	9001	555.3	688.7	560.2	467.7	178.7	226.5	234.3	143.6
220	347.1	3938	379.3	529.1	529.1	340.4	102.1	134.6	134.6	89.1
221	11634.9	1022407	15379.9	17102.7	16563.6	14667.1	1034.8	1154.0	1260.3	893.8
222	1390.9	60803	1695.9	2019.7	1704.0	1433.0	244.6	303.6	341.5	191.4
223	1815.3	99088	2296.8	2721.7	2331.6	1958.2	263.3	325.2	366.4	206.5
224	379.6	5019	388.7	505.8	442.5	313.4	144.5	186.1	206.9	110.9
225	494.9	7289	518.6	656.3	580.7	424.4	173.2	219.7	251.4	135.6
226	507.4	8046	539.8	680.6	605.8	451.1	167.3	212.9	247.9	137.1
227	687.1	10994	791.0	972.4	861.7	678.2	168.0	212.9	213.3	142.9
228	677.2	17002	786.1	955.4	861.1	663.6	155.8	199.4	200.1	520.4
229	682.4	17456	793.3	963.9	882.0	661.4	155.8	199.4	199.1	118.3
230	890.6	26123	991.1	1206.6	1067.6	854.5	238.4	296.4	305.1	197.0
231	2263.8	61558	2553.6	3067.0	2704.2	2273.7	557.4	648.2	703.9	499.6
232	2312.6	69102	2634.7	3166.6	2771.6	2349.4	541.0	632.2	707.9	481.2
233	2562.2	85663	2976.0	3548.5	3114.6	2656.3	541.0	632.2	707.9	481.2
234	2256.5	117486	2699.5	3216.5	2675.7	2376.6	413.6	498.0	632.9	354.2

OBS	FILENAME	AUTH	VER	NCSL	SBL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
235	GTORDTXT.C	J	1.4	37	32	34	15	5	20	47	119	80	199	347.5	1207.2	20547	1326.2	1647.5	1492.1	1170.1	347.5	413.9	437.5	305.8
236	GTORDTXT.C	U	1.6	38	32	38	15	5	20	49	123	82	205	367.6	1252.2	20956	1375.2	1711.9	1554.8	1213.2	361.6	429.0	452.2	319.7
237	GTSHFORM.C	J	1.0	39	26	27	16	4	21	49	124	82	206	367.4	1262.6	22186	1383.6	1730.4	1593.1	1235.1	367.4	436.6	450.6	329.1
238	GTXTLINE.C	U	1.4	13	11	15	6	3	15	17	42	35	77	128.1	385.0	5945	406.0	521.3	467.3	363.5	128.1	166.5	169.4	104.0
239	HELP.C	U	1.4	39	22	16	20	7	20	33	117	73	190	252.9	1088.3	24075	1255.7	1528.8	1363.4	1063.6	252.9	310.8	343.4	213.2
240	HELP.C	U	1.6	43	25	18	21	9	20	33	127	78	205	252.9	1174.2	27754	1377.8	1675.1	1484.5	1166.2	252.9	310.8	343.4	213.2
241	HISTARCH.C	H	1.4	64	27	23	32	13	24	59	238	167	405	457.1	2581.9	87697	3112.0	3370.0	3238.0	2836.0	457.1	569.3	623.8	391.1
242	HISTINIT.C	H	1.4	55	31	39	34	18	28	44	211	143	354	374.8	2184.2	99379	2653.0	2862.9	2578.9	2327.4	374.8	523.1	603.4	317.5
243	HISTISRT.C	H	1.4	62	36	37	35	17	26	41	197	156	353	341.9	2141.3	105918	2638.0	3136.7	2747.3	2360.7	341.9	413.9	559.4	295.3
244	HISTNEXT.C	H	1.4	15	14	21	6	3	11	15	32	26	58	96.7	272.6	2599	282.2	367.2	329.6	234.5	96.7	128.4	115.7	75.6
245	HISTPREV.C	H	1.4	15	15	20	6	3	11	15	32	26	58	96.7	272.6	2599	282.2	369.2	329.6	234.5	96.7	128.4	115.7	75.6
246	HOMEOBJ.C	Q	1.0	18	20	27	10	5	18	16	60	47	107	139.1	544.4	14392	615.5	795.5	699.9	543.8	139.1	179.5	197.9	103.7
247	HSCROLL.C	Q	1.0	49	23	42	34	6	22	22	168	97	265	196.2	1446.7	70168	1882.1	2228.4	2025.0	1670.4	196.2	247.1	256.5	155.2
248	HSHRINK.C	G	1.4	44	28	34	27	8	22	33	140	115	255	264.6	1474.2	56513	1785.3	2171.2	1918.3	1615.5	264.6	325.2	397.9	226.9
249	ID2ONUM.C	U	1.0	23	8	16	13	6	18	32	69	46	115	173.2	612.0	11517	675.6	828.8	665.4	598.4	173.2	219.7	255.4	139.2
250	IMEXWIN.C	U	1.0	35	19	22	3	7	14	39	121	76	197	259.4	1128.4	15393	1312.0	1629.2	1451.6	1122.2	259.4	310.8	315.3	222.8
251	IMMARK.C	U	1.0	65	46	64	38	11	24	54	209	155	364	420.8	2287.9	78805	2738.6	3326.7	2891.7	2457.1	420.8	498.0	651.1	374.4
252	INIT.C	U	1.4	52	36	29	30	12	26	56	210	137	347	447.4	2206.1	70161	2592.4	3057.1	2608.9	2262.1	447.4	529.1	631.6	383.4
253	INIT.C	U	1.6	52	37	33	30	12	26	55	210	137	347	440.2	2199.9	71238	2592.4	3057.1	2608.9	2262.1	440.2	521.3	624.2	376.4
254	INIT.C	U	1.7	49	38	39	29	11	26	53	194	127	321	425.8	2023.5	63034	2361.9	2800.2	2396.0	2049.2	425.8	505.8	587.7	362.3
255	INITBUF.C	U	1.0	109	19	28	64	28	26	56	402	254	656	447.4	4170.6	245914	5506.8	6431.0	5566.0	4940.8	447.4	529.1	712.0	392.7
256	INITSCR.C	U	1.0	17	10	19	8	5	11	17	52	38	90	107.5	432.7	5319	495.8	600.2	510.7	444.6	107.5	140.9	135.3	92.5
257	INITVDB.C	U	1.4	21	18	30	.	7	20	24	74	64	138	196.5	753.4	20091	843.5	957.0	843.1	722.8	196.5	241.4	270.4	161.0
258	INSBLNK.C	U	1.4	64	44	46	.	16	28	47	176	132	308	395.7	1918.5	75433	2242.7	2643.5	2256.6	1958.1	395.7	474.8	589.3	340.2
259	INSDPTR.C	C	1.4	42	29	35	.	8	24	31	116	83	199	263.6	1150.5	36964	1324.7	1565.2	1345.9	1145.8	263.6	325.2	392.7	213.2
260	ITOA.C	U	1.4	17	2	16	11	4	17	10	48	33	81	102.7	385.1	10803	434.5	529.1	438.4	356.1	102.7	128.4	133.3	72.8
261	IWOUTOPT.C	G	1.0	26	21	33	13	5	16	27	75	48	123	192.4	667.4	9492	735.2	887.6	805.4	638.6	192.4	240.2	223.4	155.5
262	IWOUTOPT.C	G	1.1	26	22	35	13	6	17	27	76	49	125	197.9	682.4	10527	750.0	904.4	817.2	649.7	197.9	247.1	228.0	158.8
263	JUMPVTOD.C	B	1.4	36	29	44	.	11	22	29	91	73	164	239.0	930.3	25759	1044.1	1330.7	1091.3	879.9	239.0	296.4	345.2	195.3
264	KEYCON.C	G	1.0	12	16	26	5	4	11	11	26	19	45	76.1	200.7	1906	202.9	261.1	217.6	161.3	76.1	104.0	99.1	53.2
265	KEYFOUND.C	U	1.4	20	11	3	11	4	16	19	64	47	111	144.7	569.4	11267	645.1	778.9	670.4	548.0	144.7	179.5	200.6	109.7
266	KILLCURR.C	U	1.0	14	5	0	8	2	18	17	44	27	71	144.5	144.5	5206	368.6	459.5	407.7	335.7	144.5	186.1	186.8	118.7
267	KILLCURR.C	U	1.1	14	9	10	7	2	17	15	39	22	61	128.1	305.0	3802	304.2	391.5	342.6	270.3	128.1	166.5	167.8	106.2
268	KILLCURR.C	U	1.2	18	17	19	9	2	19	19	51	30	81	161.4	425.1	6376	436.7	552.7	497.5	377.2	161.4	206.1	212.7	129.1
269	LEFTOBJ.C	Q	1.0	23	21	28	13	7	19	19	82	61	143	161.4	750.5	22889	883.1	1119.1	971.9	772.6	161.4	206.1	239.0	124.5
270	LEXLINE.C	P	1.4	182	76	96	101	31	30	88	486	334	820	715.6	5643.8	321310	7137.6	8213.2	7759.8	6410.8	715.6	848.8	959.0	645.0
271	LEXLINE.C	P	1.6	186	74	103	101	33	31	92	503	344	847	753.7	5880.3	340802	7412.8	8548.2	8032.1	6659.6	753.7	890.5	1035.1	679.9
272	LEXUFIND.C	U	1.0	37	21	38	22	4	23	25	112	75	187	220.1	1044.4	36031	1229.6	1456.3	1300.3	1064.0	220.1	275.1	304.9	185.0
273	LFINDDEC.C	U	1.4	32	15	31	14	4	18	19	77	53	130	155.8	677.2	17002	786.1	955.4	861.1	663.6	155.8	199.4	200.1	120.4
274	LFINDPOS.C	B	1.4	31	14	33	13	5	19	19	81	67	138	161.4	724.2	20640	846.0	1023.9	909.6	717.8	161.4	206.1	217.0	123.2
275	LFINDUN.C	U	1.4	28	15	22	14	4	18	19	77	54	131	155.8	682.4	17456	793.3	963.9	882.0	661.4	155.8	199.4	199.1	118.3
276	LFINDU2.C	U	1.0	28	15	24	14	4	18	19	76	54	130	155.8	677.2	17323	785.6	955.4	873.7	658.3	155.8	199.4	199.1	118.3
277	LIMDOWN.C	Q	1.0	28	21	29	17	8	23	20	93	68	161	190.5	873.6	34159	1022.1	1286.2	1115.9	903.3	190.5	240.2	254.1	149.2
278	LIMLEFT.C	Q	1.0	28	21	29	17	8	23	20	93	68	161	190.5	873.6	34159	1022.1	1286.2	1115.9	903.3	190.5	240.2	254.1	149.2
279	LIMRIGH.C	Q	1.0	28	21	29	17	8	23	20	93	68	161	190.5	873.6	34159	1022.1	1286.2	1115.9	903.3	190.5	240.2	254.1	149.2
280	LIMUP.C	Q	1.0	28	21	29	17	8	23	20	93	68	161	190.5	873.6	34159	1022.1	1286.2	1115.9	903.3	190.5	240.2	254.1	149.2
281	LINEDIT.C	U	1.4	167	77	90	105	54	33	70	500	379	879	595.5	5877.4	525065	7729.4	8877.4	7682.5	6852.7	595.5	696.8	1115.1	577.8
282	LOADOBJ.C	Q	1.0	103	25	38	53	31	30	35	296	162	458	326.7	2758.2	191501	3619.1	4337.5	3653.9	3154.3	326.7	398.9	454.6	265.0

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
283	LOADS.C	Q	1.0	19	18	35	11	5	18	20	64	53	117	161.5
284	LOADVTOD.C	U	1.4	27	19	19	.	7	22	27	76	68	144	226.5
285	LOADW.C	Q	1.0	27	23	39	21	6	24	21	94	59	153	202.3
286	LOCATE.C	D	1.4	12	5	0	8	1	11	16	27	23	60	102.1
287	LOCOCUR.C	Q	1.0	23	21	30	14	5	20	28	84	59	143	221.0
288	LOCTCUR.C	Q	1.0	58	26	34	28	18	23	35	195	126	321	283.6
289	LOCWCUR.C	Q	1.0	28	24	33	18	7	21	26	92	61	153	214.5
290	LOGO.C	U	1.0	34	14	6	20	3	12	28	85	60	145	177.6
291	LOIMFLDS.C	G	1.0	56	27	25	39	17	21	60	266	219	485	446.7
292	LOIMFLDS.C	G	1.1	56	27	25	39	17	21	60	268	219	487	446.7
293	LONGTOA.C	U	1.4	17	2	16	11	4	18	10	48	33	81	108.3
294	LPSTATUS.C	U	1.4	12	5	0	8	1	12	15	28	28	56	101.6
295	LSEGSRCH.C	H	1.4	26	11	15	15	4	20	19	65	48	113	167.1
296	LUFIND.C	U	1.4	29	16	38	14	4	20	22	77	55	132	184.5
297	LUFIRSTF.C	A	1.4	65	29	62	33	9	27	37	174	130	304	321.1
298	LUFIRSTF.C	A	1.5	70	31	68	34	11	28	45	198	146	344	381.7
299	LUNEXTFI.C	A	1.4	42	20	37	25	6	22	26	112	82	194	220.3
300	LUNEXTFI.C	A	1.5	47	21	44	26	7	26	33	128	91	219	288.7
301	MARK.C	U	1.0	56	43	50	32	10	25	47	177	130	307	377.2
302	MARKST.C	U	1.4	491	224	151	304	113	39	157	1254	967	2221	1351.4
303	MARKTOS.C	U	1.4	32	11	13	.	6	21	18	94	54	148	167.3
304	MDOCINIT.C	G	1.0	40	19	33	27	4	23	52	149	137	286	400.5
305	MDOCSUP.C	G	1.0	94	36	46	46	11	19	37	194	169	363	273.5
306	MDREINIT.C	G	1.0	26	19	30	13	3	20	49	100	77	177	361.6
307	MESSAGE.C	U	1.4	31	10	0	19	4	21	21	75	56	131	184.5
308	MKCURINV.C	U	1.0	34	17	12	15	7	9	41	120	76	196	300.4

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
283	614.0	14644	687.6	879.1	782.8	661.5	161.5	206.1	227.9	125.1
284	888.5	22399	888.8	1127.8	917.2	761.0	226.5	282.2	330.7	183.3
285	840.3	28329	963.2	1127.8	953.2	845.8	202.3	254.1	297.1	158.7
286	285.3	3236	294.8	429.0	429.0	254.0	102.1	134.6	134.6	84.6
287	798.6	16829	884.0	1084.3	984.5	772.1	221.0	275.1	285.1	180.2
288	1880.4	77849	2362.6	2741.3	2417.5	2038.5	283.6	347.1	406.0	232.9
289	849.9	20936	961.9	1145.3	1016.7	829.8	214.5	268.1	291.7	168.7
290	771.7	9922	899.2	1110.4	1014.5	765.9	177.6	219.7	230.8	152.2
291	3074.8	117843	3845.4	4212.3	4181.3	3392.4	446.7	687.7	703.1	401.7
292	3087.5	118329	3864.4	4231.9	4200.2	3410.2	446.7	687.7	703.1	401.7
293	389.4	11565	434.5	529.1	438.4	356.1	108.3	134.6	139.2	77.7
294	266.3	.	269.0	354.0	354.0	234.5	101.6	134.6	134.6	85.3
295	597.3	15088	659.5	828.8	694.2	547.0	167.1	212.9	245.3	126.4
296	711.8	17795	800.5	972.4	839.5	684.5	184.5	233.3	248.2	151.3
297	1824.0	86517	2208.0	2604.5	2248.6	1972.7	321.1	391.5	497.1	271.5
298	2129.3	96718	2560.3	3017.4	2588.1	2281.4	381.7	459.5	578.2	328.6
299	1083.5	37589	1283.7	1537.9	1334.4	1122.5	220.3	275.1	352.1	181.7
300	1288.3	46184	1488.2	1795.2	1545.4	1301.4	288.7	354.4	422.5	243.1
301	1894.2	65490	2234.7	2711.9	2321.0	1996.7	377.2	451.9	579.5	329.2
302	16912.3	2031250	22496.7	25511.9	22774.9	20608.9	1351.4	1508.0	2108.0	1265.3
303	782.2	24641	926.9	1136.5	947.4	780.2	167.3	212.9	260.0	132.1
304	1781.4	53974	2048.1	2487.5	2299.3	1752.6	400.5	482.7	529.0	340.3
305	2108.1	91473	2725.1	3258.3	3043.0	2513.6	273.5	350.1	450.0	230.2
306	1081.2	16990	1146.9	1386.0	1326.2	972.5	361.6	437.0	450.1	310.1
307	706.4	19779	792.4	1049.7	937.1	691.0	184.5	233.3	285.0	148.8
308	1157.8	20388	1303.7	1562.0	1369.0	1156.4	300.4	374.1	375.2	255.1

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
309	MKCURINV.C	U	1.1	34	18	15	15	7	9	40	119	75	194	293.6
310	MKCURINV.C	U	1.4	55	26	27	26	13	24	65	209	133	342	501.5
311	MKWKHIST.C	U	1.4	97	42	35	86	10	24	61	338	299	637	471.8
312	MODECO80.C	U	1.4	10	5	3	6	1	10	13	22	22	44	81.3
313	MOVE.C	U	1.4	12	12	13	7	1	12	12	32	24	566	86.0
314	MOVEW.C	Q	1.0	113	80	116	62	43	26	37	467	314	781	315.0
315	MSGPRT.C	U	1.4	98	42	63	57	17	29	81	304	201	505	654.4
316	MSGPRT.C	U	1.5	102	45	66	61	17	29	81	315	205	520	654.4
317	MSTXT.C	G	1.0	31	21	37	14	8	23	29	86	58	144	244.9
318	MVDIR.C	U	1.4	39	15	0	32	2	19	31	130	101	231	234.3
319	NBLOCK.C	G	1.4	27	23	28	12	3	20	30	95	70	165	233.6
320	NEWINFO.C	U	1.4	10	8	10	.	2	13	8	18	14	32	72.1
321	NEWTOKEN.C	H	1.4	171	59	55	77	48	25	79	450	329	779	614.1
322	NOISE.C	U	1.4	70	47	59	46	15	30	51	276	196	472	436.5
323	NOMVDISP.C	U	1.4	14	19	27	.	2	15	18	37	29	66	133.7
324	NOTEDEL.C	B	1.0	49	26	18	26	8	23	39	109	71	180	310.2
325	NOTSEGEX.C	H	1.4	15	11	19	7	4	14	14	34	24	58	106.6
326	NOTWOP.C	A	1.4	170	32	92	63	21	24	36	320	208	528	296.2
327	NOTWOP.C	A	1.6	200	39	100	83	25	31	51	427	278	705	442.9
328	NOTWSEGO.C	B	1.4	144	29	90	53	20	24	35	273	183	456	289.6
329	NOTWSEGO.C	B	1.6	162	35	98	65	22	30	50	336	223	559	429.5
330	NPFULL.C	B	1.4	275	141	137	152	62	34	144	689	519	1208	1205.4
331	NPHALF.C	B	1.4	324	157	152	176	69	37	158	829	634	1463	1346.7
332	NPPRINT.C	U	1.4	36	28	37	16	6	19	43	110	81	191	314.0
333	NPPRINT.C	U	1.6	39	51	42	19	7	19	44	119	88	207	320.9
334	NPWRITE.C	D	1.4	242	103	111	145	46	38	143	715	518	1233	1223.3

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
309	1141.2	20328	1287.6	1543.7	1352.1	1140.9	293.6	366.7	368.3	248.5
310	2214.7	54379	2549.2	2991.1	2651.7	2292.2	501.5	596.0	614.9	439.4
311	4082.8	240148	5298.5	6030.7	5625.3	4692.2	471.8	552.7	607.2	413.6
312	199.0	1684	196.2	282.2	282.2	170.9	81.3	110.0	110.0	68.6
313	256.8	3081	270.0	347.1	347.1	218.6	86.0	116.1	116.1	63.2
314	4668.3	515022	6745.5	7670.8	6881.5	5964.3	315.0	384.0	495.4	263.5
315	3424.6	123222	4045.2	4724.9	4129.0	3604.4	654.4	761.5	884.6	586.9
316	3526.3	129407	4188.5	4882.1	4276.6	3731.8	654.4	761.5	891.3	583.4
317	820.9	18880	892.4	1110.4	917.7	745.8	244.9	303.6	328.2	194.4
318	1303.7	40353	1585.4	1753.1	1794.3	1506.5	234.3	291.9	334.8	200.8
319	931.2	21729	1053.2	1230.1	1191.5	893.8	233.6	313.2	301.9	197.5
320	140.6	1599	128.4	173.0	152.5	97.7	72.1	98.1	94.4	50.4
321	5219.6	271719	6717.3	8026.2	7198.8	6092.9	614.1	705.0	852.8	561.7
322	2992.4	172504	3730.4	4225.8	3756.0	3295.7	436.5	533.2	675.1	367.3
323	332.9	4023	333.6	421.5	388.8	277.7	133.7	173.0	169.1	104.0
324	1071.8	22438	1174.4	1456.3	1453.3	1037.6	310.2	376.6	376.6	266.4
325	278.8	3346	283.0	369.2	473.7	226.1	106.6	140.9	150.0	77.4
326	311.8	216240	4264.7	5006.8	4482.4	3977.3	296.2	361.8	424.3	251.5
327	4482.1	378691	5988.2	6976.5	6315.1	5553.6	442.5	529.1	596.5	377.4
328	2682.5	168307	3584.7	4213.3	3747.1	3334.6	289.6	354.4	404.3	245.3
329	3534.0	236422	4559.4	5335.0	4796.0	4220.9	429.5	513.5	569.2	366.6
330	9030.7	553318	11177.3	13024.0	11768.3	10261.4	1205.4	1339.6	1637.3	1103.8
331	11129.5	826191	13938.8	16198.4	14751.9	12828.7	1346.7	1492.5	1818.9	1227.5
332	1137.3	20352	1259.5	1547.0	1375.1	1094.4	314.0	376.6	369.3	273.4
333	1237.3	23509	1388.9	1693.5	1508.2	1210.7	320.9	384.0	383.6	280.2
334	9247.3	636448	11450.2	12810.4	12095.9	10524.5	1223.3	1443.9	1628.0	1128.9

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
335	NPWRITE.C	D	1.5	242	104	111	145	46	38	143	715	518	1233	1223.3
336	NSETEMPH.C	A	1.4	32	22	45	15	12	27	27	105	71	176	256.8
337	NUMOBJS.C	U	1.0	21	7	14	11	6	18	20	67	45	112	161.5
338	OPENALL.C		1.4	45	24	45	.	12	23	30	136	104	240	251.2
339	OPENALL.C		1.6	55	33	61	.	13	23	34	156	119	275	277.0
340	OPENALL.C		1.7	57	34	63	.	13	23	38	164	123	287	303.5
341	OPENALL.C		1.8	58	34	63	.	13	23	39	169	126	295	310.2
342	OPENS.C	Q	1.0	116	44	49	78	23	31	81	449	299	748	667.1
343	OPENSF.C	Q	1.0	96	233	4	5	1	2	4	37	23	60	375.0
344	OPENV.C	U	1.4	23	16	42	.	4	14	14	54	39	93	106.6
345	OPENW.C	Q	1.0	79	35	38	45	19	27	63	302	210	512	505.0
346	ORDER.C	J	1.0	52	50	76	28	15	26	42	152	98	250	348.7
347	ORDER.C	J	1.1	145	101	135	68	43	34	72	375	248	623	617.2
348	ORDERFO.C	B	1.4	49	17	14	18	10	18	31	117	82	199	228.6
349	ORDERFOK.C	B	1.0	49	24	28	26	8	23	39	109	71	180	310.2
350	ORDERPR.C	U	1.0	17	8	9	4	4	13	26	71	40	111	170.3
351	OREXPR.C	H	1.4	16	9	16	7	5	14	14	37	26	63	106.6
352	OROP.C	U	1.4	134	24	73	58	21	22	27	307	200	507	226.5
353	OROP.C	U	1.6	168	31	90	75	27	29	43	401	259	660	374.2
354	OUTITEM.C	G	1.0	342	145	175	191	76	40	165	1029	743	1772	1428.3
355	OUTITEM.C	G	1.1	342	145	178	191	76	40	165	1028	743	1771	1428.3
356	OUTITEM.C	G	1.2	307	140	172	161	61	39	155	847	615	1462	1421.6
357	OUTITEM.C	G	1.3	313	143	175	165	62	39	169	868	626	1494	1456.9
358	OUTPOPT.C	M	1.0	498	132	111	306	122	42	146	1345	1009	2354	1276.2
359	OUTPOPT.C	M	1.1	506	158	127	317	144	42	146	1447	1088	2535	1276.2
360	OUTPOPT.C	M	1.2	558	177	139	351	154	43	161	1577	1180	2757	1413.6

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
335	9247.3	636448	11450.2	12810.4	12095.9	10524.5	1223.3	1443.9	1628.0	1128.9
336	1012.9	35957	1141.6	1384.3	1123.2	993.9	256.8	318.0	342.3	213.3
337	587.8	11902	653.6	787.2	680.0	573.7	161.5	206.1	209.7	132.4
338	1374.7	54805	1660.7	2029.1	1644.5	1461.6	251.2	310.8	387.5	203.1
339	1604.0	64563	1957.0	2401.1	1972.5	1426.7	277.0	339.8	426.7	228.1
340	1702.1	63359	2060.6	2536.5	2082.4	1815.3	303.5	369.2	452.6	253.9
341	1756.5	65260	2129.9	2624.0	2154.7	1873.3	310.2	376.6	459.0	260.4
342	5091.9	291339	6414.9	7095.2	6548.6	5682.3	667.3	840.6	1026.0	558.4
343	3720.5	257828	990.3	206.4	5236.2	4443.9	375.5	571.2	593.5	313.7
344	447.1	8718	516.9	640.2	550.5	440.1	106.6	140.9	158.1	718.5
345	3323.8	149592	4108.0	4838.4	4224.2	3588.4	505.0	592.2	682.6	412.5
346	1521.9	46163	1749.9	2123.7	1791.9	1529.0	348.7	421.5	495.1	292.0
347	4191.5	245435	5179.2	6203.4	5418.1	4631.8	617.2	721.3	916.8	555.4
348	1117.3	26600	1325.2	1434.2	1368.6	1160.6	228.6	313.9	297.6	195.5
349	1071.8	22438	1174.4	1456.3	1456.3	1037.6	310.2	376.6	376.6	266.4
350	586.7	5867	649.5	828.8	738.3	558.5	170.3	212.9	208.1	139.4
351	302.9	3927	315.0	406.4	802.6	253.8	106.6	140.9	160.3	77.4
352	2846.7	231950	4065.2	4785.9	4302.3	3827.5	226.5	282.2	310.0	187.4
353	4072.2	355650	5544.0	6507.0	5877.0	5184.6	374.2	457.9	487.6	315.0
354	13608.0	1225548	17383.4	19832.9	18207.6	16062.0	1428.3	1583.4	1946.4	1335.2
355	13600.4	1224857	17371.9	19820.6	18195.8	16050.9	1428.3	1583.4	1946.4	1335.2
356	11217.1	815278	13935.7	16066.2	14706.3	12824.3	1421.6	1574.3	1893.9	1335.6
357	11504.5	830976	14288.6	16463.1	15048.2	13159.3	1456.9	1610.8	1951.1	1370.7
358	17783.5	2580922	24047.6	27088.6	24893.9	22235.7	1276.2	1429.3	1809.8	1158.5
359	19150.9	2996811	26167.0	29329.0	26921.1	24043.7	1276.0	1429.3	1809.8	1158.5
360	21152.9	3333220	28793.8	32256.9	29492.6	26468.2	1413.6	1574.3	2032.5	1293.4

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
361	OUTPOPT.C	M	1.3	559	180	141	352	154	43	162	1579	1181	2760	1422.4
362	PACK.C	G	1.0	19	13	22	13	2	17	17	54	35	89	139.0
363	PAGEWITH.C	U	1.4	112	26	35	.	26	22	35	253	225	478	277.6
364	PARSERR.C	H	1.4	148	24	25	55	75	33	108	471	415	886	896.0
365	PAUSEPR.C	C	1.4	39	21	29	.	9	26	41	127	85	212	341.9
366	PCKEYS.C	U	1.4	34	6	1	.	1	10	39	284	251	535	239.3
367	PDBTITLE.C	G	1.0	18	6	0	9	2	16	23	60	40	100	168.0
368	PDOCSCR.C	M	1.0	198	49	50	101	65	33	92	554	388	942	766.6
369	PDOCSCR.C	M	1.2	194	46	50	99	63	32	90	536	376	912	744.3
370	PHRASEOP.C	B	1.4	204	45	38	73	27	24	42	341	247	588	336.5
371	PHRASEOP.C	B	1.6	216	48	46	87	28	32	57	382	272	654	492.5
372	PLTOSIN.C	U	1.4	30	11	10	10	16	20	19	86	64	150	167.1
373	PREOP.C	A	1.4	181	34	84	69	22	25	36	345	228	573	302.2
374	PREOP.C	A	1.6	221	43	92	89	26	32	51	452	298	750	449.3
375	PRINTSUP.C	D	1.4	147	114	139	69	26	31	110	407	312	719	899.5
376	PRINTSUP.C	D	1.6	192	137	161	89	33	34	133	469	350	819	1111.3
377	PRIOW.C	Q	1.0	73	51	71	45	15	25	30	256	157	413	263.3
378	PROMPT.C	U	1.4	58	21	22	32	10	23	50	163	112	275	386.2
379	PROMPT.C	U	1.7	58	23	32	32	10	23	50	164	113	277	386.2
380	PRSDOC.C	J	1.0	8	13	17	5	1	5	6	12	10	22	27.1
381	PRSHORT.C	J	1.0	224	80	70	116	43	28	74	496	363	859	594.1
382	PRSHORT.C	J	1.2	167	79	78	90	32	29	64	380	275	655	524.9
383	PULLWORD.C	A	1.4	52	23	51	28	5	25	42	123	86	209	342.6
384	PUTITLE.C	D	1.4	24	22	28	16	5	22	32	111	83	194	258.1
385	QAPPEND.C	G	1.4	41	24	33	23	4	21	32	132	116	246	252.2
386	QUICKIO.C		1.4	36	15	22	.	6	18	19	79	56	135	155.8

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
361	21195.4	3322112	28829.6	32295.6	29529.9	26503.0	1422.4	1583.4	2041.5	1302.1
362	452.8	7924	490.3	616.1	522.6	401.1	139.0	179.5	196.8	104.1
363	2788.1	197160	377.8	4744.0	4148.5	3316.4	277.6	339.8	494.3	231.3
364	6325.6	401063	7791.5	9522.0	8262.3	7151.8	896.0	1015.3	1190.0	838.7
365	1286.0	34660	1432.4	1614.7	1485.4	1291.8	341.9	415.1	415.2	286.5
366	3003.9	96663	4315.4	4798.2	4869.9	4057.3	239.3	248.2	282.2	208.0
367	528.5	7354	553.2	721.3	647.9	467.4	167.6	212.9	212.1	142.0
368	6561.8	456614	8385.8	9692.4	8618.4	7511.8	766.6	879.1	1025.5	684.5
369	6320.8	422513	8075.9	9351.9	8313.6	7232.1	744.3	853.9	973.9	664.7
370	3554.1	250818	4838.3	5601.4	5036.0	4442.0	336.5	406.4	499.9	289.2
371	4235.1	323356	5476.4	6344.2	5739.1	5029.3	492.5	584.3	682.0	422.6
372	792.8	26705	936.7	1154.0	1011.7	806.1	167.1	212.9	226.1	130.9
373	3398.3	269033	4694.4	5473.3	4948.8	4349.8	302.2	369.2	408.8	255.4
374	4781.3	447003	6436.0	7460.6	6805.8	5955.3	449.3	537.0	580.3	381.6
375	5133.3	225680	6133.3	6823.6	6299.3	5559.8	899.5	1086.1	1235.3	816.5
376	6047.3	270535	7199.6	8017.1	7470.5	6439.9	1111.3	1267.5	1445.4	1018.2
377	2387.7	156196	3193.3	3802.5	3361.2	2914.0	263.3	325.2	419.7	213.8
378	1702.2	43849	1960.3	2362.6	2088.5	1720.8	386.2	459.5	515.4	343.6
379	1714.6	44562	1977.3	2381.8	2106.5	1737.2	386.2	459.5	515.4	343.6
380	76.1	317	76.2	104.0	104.0	71.3	27.1	38.1	38.1	23.5
381	5731.6	393622	7528.2	8866.2	7755.7	6791.4	594.1	688.7	945.6	533.7
382	4283.1	266860	5484.9	6452.7	5637.6	4907.1	524.9	616.1	796.1	467.3
383	1267.8	32450	1406.6	1711.9	1544.5	1227.4	342.6	413.9	419.7	303.3
384	1116.4	31854	1283.3	1556.1	1386.8	1096.7	258.1	318.0	340.3	214.0
385	1420.5	54069	1725.4	1699.3	1756.5	1551.5	252.2	367.7	381.8	208.7
386	703.3	18655	823.2	947.5	891.1	715.8	155.8	221.3	237.0	122.3

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
387	QUITSRCH.C	U	1.40	49	29	54	14	7	18	38	107	75	182	274.5
388	QUITSRCH.C	U	1.60	52	31	59	16	8	19	40	115	79	194	293.6
389	RBLOCK.C	G	1.40	19	17	23	9	1	14	25	72	51	123	169.4
390	READDR.C	U	1.40	23	11	18	.	3	17	24	172	46	118	179.5
391	READDR.C	U	1.60	36	19	24	.	6	24	37	114	83	197	302.8
392	READSTXT.C	U	1.40	36	25	31	32	5	22	36	154	112	266	284.2
393	READSTXT.C	U	1.60	43	24	33	32	8	25	40	194	129	323	329.0
394	READV.C	U	1.40	17	10	11	.	4	12	23	53	48	101	147.1
395	READWD.C	Q	1.00	16	18	30	8	4	19	17	53	33	86	150.2
396	RESTART.C	B	1.00	49	25	21	26	8	23	39	109	71	180	310.2
397	RESTCOPY.C	U	1.40	69	47	50	37	16	29	62	233	170	403	510.2
398	RESTSCR.C	Q	1.00	21	19	22	11	3	14	15	63	29	92	111.9
399	RETITLE.C	G	1.00	18	20	28	9	3	17	22	59	39	98	167.6
400	REVERSE.C	U	1.40	10	3	5	7	2	12	8	31	26	57	67.0
401	REVHLITE.C	U	1.40	18	16	34	.	7	17	13	51	38	89	117.6
402	REXIT.C	B	1.00	49	25	21	26	8	23	39	109	71	180	310.2
403	RIGHOBJ.C	Q	1.00	23	21	28	13	7	19	19	82	61	143	161.4
404	SAVECOPY.C	U	1.40	69	50	41	41	12	26	67	235	176	411	528.6
405	SAVESCR.C	Q	1.00	26	23	23	16	3	18	19	119	50	169	155.8
406	SCOMP.C	G	1.00	18	14	23	10	4	17	11	44	26	70	107.5
407	SCRDUMP.C	B	1.40	21	13	18	11	2	19	17	43	35	78	150.2
408	SCRDUMP.C	B	1.50	22	13	18	11	2	19	19	47	37	84	161.4
409	SCRNOFF.C	Q	1.00	15	18	27	8	4	15	16	43	31	74	122.6
410	SCRNON.C	Q	1.00	15	18	27	8	4	15	16	43	31	74	122.6
411	SCRPLAY.C	U	1.00	42	16	8	21	7	15	38	112	78	190	258.0
412	SCRW.C	Q	1.00	121	76	118	66	31	30	39	410	246	656	353.3

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
387	1056.9	18775	1188.5	1492.5	1294.7	1057.1	274.5	332.5	360.6	234.2
388	1141.2	21412	1285.2	1601.7	1389.9	1142.3	293.6	354.4	392.9	250.8
389	650.1	9284	733.5	848.4	921.4	611.5	169.4	226.0	212.9	141.3
390	632.2	10299	698.3	819.5	767.7	596.5	179.5	228.4	235.0	147.7
391	1168.4	31451	1308.1	1543.7	1363.3	1120.7	302.8	369.9	423.3	255.9
392	1558.2	53326	1881.5	2162.8	2660.9	1654.7	284.2	368.3	382.9	233.6
393	1945.2	78417	2378.8	2720.2	2370.7	2100.8	329.0	420.0	526.6	270.6
394	518.1	6487	571.7	721.3	631.1	487.3	147.1	186.1	203.0	122.2
395	444.6	8199	470.0	584.3	521.3	399.8	150.2	192.7	196.6	116.2
396	1071.8	22438	1174.4	1456.3	1456.3	1044.2	310.2	376.6	376.6	266.4
397	2622.6	104271	3092.0	3483.4	3225.9	2810.1	510.0	629.3	770.1	446.8
398	446.9	6049	517.5	624.1	595.9	464.6	111.9	147.2	142.6	88.1
399	518.0	7805	553.2	721.3	647.9	467.4	167.6	212.9	212.1	142.0
400	246.3	4804	275.8	347.1	298.8	234.1	67.0	86.4	107.5	45.2
401	436.7	10851	488.7	600.2	516.4	420.4	117.6	153.6	168.6	86.6
402	1071.8	22438	1174.4	1456.3	1456.3	1044.2	310.2	376.6	376.6	266.4
403	750.5	22889	883.1	1119.1	971.9	772.6	161.4	206.1	239.0	124.5
404	2687.6	91779	3163.8	3513.4	3298.6	2882.8	528.6	675.4	772.9	482.2
405	880.4	20852	1102.7	1277.3	1200.5	974.4	155.8	199.4	205.8	119.6
406	336.5	6761	362.4	459.5	360.8	293.0	107.5	140.9	158.2	78.6
407	403.3	7887	412.9	576.3	548.6	364.4	150.2	192.7	181.0	120.7
408	440.8	8155	453.8	632.1	603.8	398.7	161.4	206.1	194.1	731.5
409	366.6	5327	386.9	498.0	438.6	330.5	122.6	160.0	159.1	92.8
410	366.6	5327	386.9	498.0	438.6	330.5	122.6	160.0	159.1	92.8
411	1088.3	16754	1252.7	1537.9	1337.3	1071.9	258.0	310.8	354.9	228.1
412	4007.2	379142	5512.4	6268.3	5567.9	4940.3	353.3	429.0	550.2	291.0

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
413	SDISP.C	U	1.00	23	22	33	16	3	20	20	65	44	109	172.9
414	SDSEL.C	U	1.40	699	262	193	422	134	44	277	1839	1557	3396	2487.7
415	SDSEL.C	U	1.50	722	278	246	420	145	45	301	1918	1579	3497	2725.5
416	SDSEL.C	U	1.80	729	282	252	424	147	45	305	1937	1591	3528	2764.2
417	SDSEL.C	U	1.12	726	287	263	421	147	46	303	1931	1584	3518	2744.8
418	SEGEXPR.C	H	1.40	15	6	16	7	4	14	14	34	24	58	106.6
419	SELECT.C	A	1.00	110	71	123	63	17	26	67	326	221	547	528.6
420	SELOBJ.C	Q	1.00	32	22	31	12	7	20	19	72	54	126	167.1
421	SELRBN.C	D	1.40	207	124	134	120	42	39	176	636	471	1107	1519.0
422	SELST.C	U	1.40	207	113	154	124	51	28	72	523	405	928	578.8
423	SELST.C	U	1.60	193	116	162	108	53	28	73	463	343	806	586.5
424	SELST.C	U	1.80	194	116	162	108	54	28	74	465	344	809	594.1
425	SELST.C	U	1.40	194	117	165	108	54	28	73	466	345	811	586.5
426	SENULL.C	U	1.40	673	246	193	424	95	52	315	1941	1657	3598	2910.7
427	SENULL.C	U	1.50	702	300	318	445	100	53	323	2034	1727	3761	2995.9
428	SENULL.C	U	1.60	702	300	318	445	100	53	323	2034	1727	3761	2995.9
429	SENULL.C	U	1.90	714	307	330	452	104	53	331	2088	1767	3855	3074.3
430	SETALLBM.C	G	1.00	29	21	32	19	6	20	34	85	68	153	259.4
431	SETEMPHA.C	A	1.40	20	19	36	.	8	18	13	56	40	96	123.2
432	SETVEMPH.C	D	1.40	40	30	44	.	14	27	44	162	118	280	368.6
433	SHOWTEXT.C	U	1.40	215	61	45	132	40	32	78	481	378	859	650.3
434	SHOWTEXT.C	U	1.60	225	71	68	137	43	35	79	502	392	894	677.5
435	SHTOWH.C	Q	1.00	13	15	24	6	4	14	14	38	25	63	106.6
436	SHUTDOWN.C	U	1.40	13	16	21	8	1	13	18	34	21	55	123.2
437	SHUTDOWN.C	U	1.60	16	18	24	9	2	15	22	43	27	70	156.7
438	SHUTDOWN.C	U	1.80	14	19	30	7	2	11	16	32	19	57	102.1

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
413	580.1	12762	631.7	762.4	691.5	535.9	172.9	219.7	219.9	139.0
414	28276.6	3496697	36454.7	41774.6	38369.8	33589.9	2487.7	2682.6	3351.3	2351.3
415	29495.9	3481447	37693.1	43020.4	39223.1	34692.2	2725.5	2928.3	3726.5	2585.3
416	29815.9	3499455	38072.7	43432.0	39601.2	35051.8	2764.2	2967.8	3773.8	2625.1
417	29676.9	3490714	37941.0	43259.3	39422.0	34898.5	2744.8	2948.0	3742.8	2605.9
418	278.8	3346	283.0	369.2	273.7	226.1	106.6	140.9	150.0	77.4
419	3576.9	153381	4442.8	4968.7	4575.9	4022.7	528.6	645.8	669.0	470.5
420	666.0	18927	775.9	946.9	826.6	654.1	167.1	212.9	233.1	127.8
421	8577.2	447601	10105.3	11329.6	10958.3	9193.2	1519.0	1742.4	1886.2	1396.3
422	6165.5	485533	8231.1	9533.3	8288.9	7546.8	578.8	672.5	793.4	522.8
423	5366.5	353014	6988.6	8160.0	7106.8	6372.1	586.5	680.6	777.0	530.4
424	5398.0	351307	7019.0	8193.4	7138.5	6401.7	594.1	688.7	784.9	538.0
425	5399.8	357275	7039.2	8215.8	7159.7	6421.3	586.5	680.6	777.0	530.4
426	30653.7	4192446	38219.2	45255.5	42561.8	36315.5	2910.7	3136.7	3674.1	2754.9
427	32173.8	4558670	40926.1	47487.8	44703.7	38205.9	2995.9	3226.5	3778.9	2838.0
428	32173.8	4558670	40926.1	47487.8	44703.7	38205.9	2995.9	3226.5	3778.9	2838.0
429	33095.0	4681847	42087.0	48762.3	45756.6	39263.3	3074.3	3306.7	3895.6	2912.4
430	880.5	17610	958.7	1206.6	1016.1	807.4	259.4	318.0	335.4	222.9
431	475.6	13171	538.1	656.3	563.5	469.2	123.2	160.0	174.7	96.9
432	1721.9	62342	2001.2	2228.3	1972.1	1794.3	368.6	460.1	471.4	299.7
433	5825.2	451676	7522.2	8199.2	7915.5	6852.0	650.3	754.2	972.3	581.1
434	6108.6	530443	7880.7	9329.2	8256.7	7170.7	677.5	787.2	1043.7	604.5
435	302.9	3786	315.5	406.4	351.7	269.7	106.6	140.9	135.3	80.8
436	272.5	2066	265.2	339.8	339.8	231.6	123.2	160.0	160.0	101.2
437	364.7	3357	361.7	459.5	433.1	314.0	156.7	199.4	194.1	131.2
438	242.5	1584	240.7	318.0	293.8	207.2	102.1	134.6	130.2	87.4

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
439	SHUTDOWN.C	U	1.10	13	19	31	6	2	11	15	30	18	48	96.7
440	SIDEWAYS.C	B	1.00	241	91	125	111	26	40	127	523	399	922	1100.4
441	SIDEWAYS.C	B	1.10	251	101	137	118	28	43	138	560	424	984	1214.3
442	SIDEWAYS.C	B	1.20	250	101	139	118	28	42	138	559	424	983	1207.5
443	SIMGPROC.C	D	1.00	52	24	24	23	10	24	75	186	131	317	577.2
444	SITOSN.C	Q	1.00	15	18	31	8	6	15	15	49	34	83	117.2
445	SIZEW.C	Q	1.00	167	86	130	84	55	27	42	518	400	980	354.9
446	SMSGPRT.C	U	1.00	34	20	26	22	8	25	35	100	72	172	295.6
447	STARTUP.C	U	1.40	117	66	56	71	17	30	87	297	212	509	707.7
448	STARTUP.C	U	1.50	146	81	72	87	23	35	108	393	282	675	909.1
449	STARTUP.C	U	1.60	146	81	72	87	23	35	108	393	282	675	909.1
450	STARTUP.C	U	1.80	139	83	80	82	22	35	102	361	258	619	860.1
451	STDSUP.C	U	1.00	527	192	133	345	87	43	155	1721	1248	2969	1361.1
452	STDSUP.C	U	1.10	535	191	147	357	88	43	154	1744	1268	3012	1352.4
453	STDSUP.C	U	1.30	589	224	186	373	107	40	163	1859	1351	3210	1465.9
454	STDSUP.C	U	1.40	590	225	189	373	107	48	166	1869	1357	3226	1492.3
455	STDTXT.C	U	1.40	569	167	207	345	89	48	274	1576	1238	2814	2486.9
456	STDTXT.C	U	1.50	609	193	252	360	106	48	273	1668	1295	2963	2477.4
457	STDTXT.C	U	1.60	669	210	284	383	129	49	290	1775	1364	3139	2647.3
458	STDTXT.C	U	1.70	665	210	284	381	128	49	290	1768	1360	3128	2647.3
459	STDTXT.C	U	1.80	576	197	286	340	101	44	261	1577	1215	2792	2335.5
460	STDTXT.C	U	1.90	632	214	318	361	123	45	278	1677	1280	2957	2504.2
461	STDTXT.C	U	1.10	629	217	324	358	123	45	277	1669	1272	2941	2494.6
462	STDTXT.C	U	1.11	631	218	326	359	124	45	279	1676	1276	2952	2513.8
463	STDTXT.C	U	1.12	636	217	316	359	124	45	282	1683	1280	2963	2942.5
464	STOCSEL.C	A	1.40	708	260	410	294	142	47	249	1535	1215	2750	2243.1

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
439	225.6	1489	222.3	296.4	272.7	189.7	96.7	128.4	124.1	82.1
440	6807.8	427764	8170.5	9442.5	8687.1	.	1100.4	1241.9	1418.3	991.8
441	7379.8	487498	8813.0	10148.0	9360.5	.	1214.3	1366.4	1554.2	1102.3
442	7364.5	475770	8802.5	10137.1	9349.3	8093.2	1207.5	1357.5	1545.4	1098.2
443	2101.5	4405	2323.7	2669.6	2500.5	2101.7	577.2	673.4	694.3	513.6
444	407.3	6924	448.1	506.5	479.7	379.8	117.2	153.6	162.1	90.6
445	5986.4	769674	8781.9	10022.8	9034.3	7792.8	354.9	429.0	565.8	297.0
446	1016.0	26125	1108.6	1262.3	1179.1	977.3	295.6	371.1	392.3	244.3
447	3497.0	127822	4078.0	4827.9	4383.9	3600.7	707.7	812.2	916.1	635.4
448	4832.9	220837	5682.4	6670.3	6073.5	5048.0	909.1	1032.5	1164.8	827.4
449	4832.9	220837	5682.4	6670.3	6073.5	5048.0	909.1	1032.5	1164.8	827.4
450	4393.7	194485	5733.9	6073.8	5541.2	4541.2	860.1	981.0	1119.6	783.0
451	22651.6	3921204	31335.3	35159.3	32011.5	28952.3	1361.1	1519.7	2337.1	1266.2
452	22795.6	4064095	31850.7	35732.3	32440.9	29433.8	1352.4	1510.6	2342.4	1257.6
453	24784.7	4930184	34239.5	38426.8	34876.2	31693.5	1465.9	1638.3	2514.7	1361.1
454	24994.0	4899713	34433.6	38650.0	35075.0	31871.1	1492.3	1665.9	2540.1	1387.3
455	23433.2	2542133	29459.3	33381.2	30766.9	27367.2	2486.9	2488.0	3050.0	2150.5
456	24671.2	2808722	31242.8	35393.6	32665.7	29018.6	2477.4	2682.6	3222.8	2338.8
457	26383.7	3040317	33362.8	37836.6	34863.3	31066.8	2647.3	2859.2	3422.1	2502.3
458	26291.3	3020778	33229.8	37692.4	34729.4	30936.7	2647.3	2859.2	3422.1	2502.3
459	23041.4	2359762	29202.2	33057.7	30429.6	27080.6	2335.5	2526.8	3045.6	2206.0
460	24646.7	2553436	31175.5	35341.6	32480.3	28984.6	2504.2	2702.1	3243.4	2368.5
461	24501.2	2531499	30984.2	35133.3	32278.2	28802.7	2494.6	2692.3	3233.8	2359.0
462	24619.2	2533399	31116.3	35276.5	32405.2	28925.4	2513.8	2711.9	3267.5	2377.9
463	24750.4	2527698	31248.5	35432.7	32546.9	29044.3	2542.5	2741.3	3293.3	2406.5
464	22576.0	2588760	28960.3	32631.2	29294.5	26642.7	2243.1	2439.2	3213.5	2109.3

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
465	STOCSEL.C	A	1.5	712	263	416	297	143	47	252	1547	1222	2769	2771.3
466	STOCSEL.C	A	1.7	727	268	423	301	147	48	257	1578	1242	2820	2325.5
467	STODISP.C	U	1.4	45	34	45	.	11	28	29	141	93	234	275.5
468	STRAPRT.C	U	1.4	9	7	9	5	2	10	7	19	15	34	52.9
469	STRCENT.C	U	1.4	19	8	7	14	4	14	13	55	42	97	101.4
470	STRDEL.C	U	1.4	20	13	8	11	3	12	17	52	44	96	112.5
471	STRINSRT.C	U	1.4	22	14	10	13	4	13	18	59	49	108	123.2
472	STRLEFT.C	U	1.4	20	7	7	12	4	12	10	49	32	81	76.2
473	STRMID.C	U	1.4	27	11	9	15	6	17	12	67	47	114	112.5
474	STROP.C	K	1.4	23	30	44	17	6	18	18	72	46	118	150.1
475	STRPAD.C	U	1.4	13	8	8	11	3	13	8	37	28	65	72.1
476	STRPAD.C	U	1.5	13	8	8	10	3	15	10	38	29	67	91.8
477	STRPRT.C	U	1.4	7	7	9	4	1	8	6	14	10	24	39.5
478	STRREP.C	U	1.4	23	14	9	14	3	14	19	65	54	119	134.0
479	STRRIGHT.C	U	1.4	25	7	6	13	5	17	11	57	39	96	107.5
480	STRTRIM.C	U	1.4	19	9	11	10	9	14	12	51	42	93	96.3
481	STRTRIM.C	U	1.5	21	13	17	11	10	16	13	58	46	104	112.1
482	STRTRIM.C	U	1.6	19	13	19	10	9	14	12	51	42	93	96.3
483	STRTRMWB.C	U	1.4	22	16	23	11	11	17	13	61	48	109	117.3
484	STXTDISP.C	U	1.4	66	25	27	.	9	30	52	179	139	318	443.6
485	STXTDISP.C	U	1.6	65	26	29	.	9	28	46	161	124	285	388.7
486	STXTDISP.C	U	1.7	60	26	34	.	7	28	46	161	124	285	388.7
487	SVDOCTXT.C	N	1.0	73	31	33	35	11	25	78	228	162	390	606.4
488	SWIMAGE.C	D	1.0	86	19	30	40	10	25	88	290	206	496	684.5
489	SWIMAGE.C	D	1.2	117	32	34	55	10	27	107	427	298	725	849.7
490	SWIMAGE.C	D	1.3	114	33	37	52	10	27	105	421	292	713	833.4

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
465	22772.3	2595044	28922.5	32876.7	29529.9	26846.7	2271.3	2468.6	3242.1	2737.3
466	23272.5	2699250	29530.3	33601.4	30137.1	27408.8	2325.5	2526.8	3345.5	2181.1
467	1364.9	61279	1614.8	1888.3	1595.4	1394.2	275.5	339.8	405.8	222.0
468	139.0	1489	139.3	179.5	156.4	116.1	52.9	69.5	64.9	37.2
469	461.2	10440	544.5	656.3	542.8	468.4	101.4	128.4	147.8	75.7
470	466.4	7242	536.6	640.2	593.8	485.2	112.5	140.9	135.7	87.0
471	535.1	9468	622.2	737.7	679.5	562.7	123.2	153.6	138.2	96.8
472	361.2	6935	435.1	529.1	421.2	361.6	76.2	98.1	136.1	54.9
473	553.8	8437	67.5	3.8	645.7	564.7	112.5	140.9	213.1	84.0
474	610.1	14031	698.3	837.2	739.4	591.9	150.1	192.7	213.3	116.8
475	285.5	6495	327.4	398.9	353.8	293.0	72.1	92.2	104.7	59.2
476	311.1	767	40.3	21.5	375.5	299.0	91.8	122.2	134.0	75.3
477	91.4	609	86.5	116.1	116.1	73.5	39.5	53.3	53.3	29.1
478	600.3	11943	702.2	828.8	769.2	635.3	134.0	166.5	150.6	105.5
479	461.5	13908	538.6	640.2	552.0	460.4	107.5	134.6	143.2	84.2
480	437.1	10710	575.8	632.2	483.0	425.2	96.3	122.2	170.9	67.3
481	505.2	14302	593.8	721.3	553.9	489.2	112.1	140.9	187.2	76.5
482	437.1	10710	515.8	632.2	483.0	425.2	96.3	122.2	170.9	67.3
483	534.9	16786	629.9	762.4	586.8	518.9	117.6	147.2	198.2	80.1
484	2021.7	81063	2329.1	2780.5	2450.1	1469.5	443.6	529.1	594.1	390.2
485	1769.7	66786	2042.6	2449.3	2179.9	1806.0	388.7	467.2	508.5	339.4
486	1769.7	66786	2042.6	2449.3	2179.9	1806.0	388.7	467.2	508.5	339.4
487	2607.7	67701	2975.0	3493.6	3135.2	2665.9	606.4	720.6	815.5	549.3
488	3382.5	98986	3955.6	4702.1	4294.7	3585.3	684.5	778.9	871.9	676.8
489	5122.9	192612	6180.5	7174.5	6707.8	5653.6	849.7	955.4	1046.7	778.7
490	5022.7	188565	6061.5	7042.2	6588.7	5540.0	833.4	938.4	1031.7	762.6

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
491	SYSERR.C	B	1.4	21	17	24	15	1	18	43	78	67	145	308.4
492	SYSERR.C	B	1.6	96	33	35	51	20	28	133	258	234	492	1073.0
493	SYSERR.C	B	1.4	21	17	24	15	1	18	43	78	67	145	308.4
494	SYSERR.C	B	1.6	96	33	35	51	20	28	133	258	234	492	1073.0
495	TARGLIST.C	A	1.4	165	54	103	66	29	34	109	406	329	735	910.7
496	TARGLIST.C	A	1.6	166	55	109	66	29	34	110	409	330	739	918.9
497	TERMSCDN.C	U	1.4	23	10	0	13	1	12	23	40	50	90	147.1
498	TERMSCDN.C	U	1.4	23	10	0	13	1	12	23	40	50	90	147.1
499	TERMSCUP.C	U	1.4	23	8	0	13	1	12	23	40	50	90	147.1
500	TERMSCUP.C	U	1.4	23	8	0	13	1	12	23	40	50	90	147.1
501	THROWAWA.C	U	1.4	33	6	0	.	7	18	19	75	63	138	155.8
502	TIMEDOS.C	U	1.4	12	5	0	8	1	13	16	40	33	73	112.1
503	TIMEDOS.C	U	1.4	12	5	0	8	1	13	16	40	33	73	112.1
504	TIMEPAUS.C	M	1.4	17	15	22	8	3	13	15	44	31	75	106.7
505	TIMEPAUS.C	M	1.4	17	15	22	8	3	13	15	44	31	75	106.7
506	TIMEWARN.C	I	1.4	71	27	24	36	17	26	55	222	139	361	440.2
507	TIMEWARN.C	I	1.4	71	27	24	36	17	26	55	222	139	361	440.2
508	TITLEPRT.C	D	1.4	63	50	70	34	10	27	64	201	147	348	512.4
509	TITLEPRT.C	D	1.6	66	52	75	37	11	27	65	210	154	364	519.8
510	TITLEPRT.C	D	1.4	63	50	70	34	10	27	64	201	147	348	512.4
511	TITLEPRT.C	D	1.6	66	52	75	37	11	27	65	210	154	364	519.8
512	TOGGOBJ.C	U	1.0	31	9	19	15	8	20	37	123	79	202	279.0
513	TOKENREC.C	U	1.4	129	42	60	70	40	24	93	290	199	489	718.2
514	TOPPRIO.C	Q	1.0	17	19	28	12	3	15	15	59	32	91	117.2
515	TUPSUP.C	B	1.0	698	316	432	476	90	42	250	2203	1738	3941	2217.9
516	TUPSUP.C	B	1.1	694	314	432	474	89	42	250	2182	1720	3902	2217.9

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
491	860.0	12059	896.7	984.4	1093.0	797.6	308.4	378.4	369.2	272.5
492	3606.8	88842	3908.6	4152.8	4065.9	3623.4	1073.0	1109.1	1105.7	1013.4
493	860.0	12059	896.7	984.4	1093.0	797.6	308.4	378.4	369.2	272.5
494	3606.8	88842	3908.6	4152.8	4065.9	3623.4	1073.0	1109.1	1105.7	1013.4
495	5262.5	270029	6269.2	7295.1	6575.0	5711.4	910.7	1032.5	1072.2	825.8
496	5298.6	270227	6309.4	7350.2	6629.3	5750.4	918.9	1041.1	1080.7	834.0
497	461.6	6021	495.1	713.2	713.2	440.9	147.1	186.1	186.1	126.4
498	461.6	6021	495.1	713.2	713.2	440.9	147.1	186.1	186.1	126.4
499	461.6	6021	495.1	713.2	713.2	440.9	147.1	186.1	186.1	126.4
500	461.6	6021	495.1	713.2	713.2	440.9	147.1	186.1	186.1	126.4
501	718.9	21454	843.7	1075.7	911.6	723.7	155.8	199.4	247.1	122.3
502	354.6	4754	379.3	529.1	529.1	329.8	112.1	147.2	147.2	91.1
503	354.6	4754	379.3	529.1	529.1	829.8	112.1	147.2	147.2	91.1
504	360.6	4843	393.8	505.8	439.1	330.9	106.7	140.9	137.7	84.1
505	360.6	4843	393.8	505.8	439.1	330.9	106.7	140.9	137.7	84.1
506	2288.7	75194	2719.0	3246.5	2851.7	2384.8	440.2	521.3	577.0	381.3
507	2288.7	75194	2719.0	3246.5	2851.7	2384.8	440.3	521.3	577.0	381.3
508	2264.7	70224	2596.2	3096.8	2793.8	2311.8	512.4	600.2	660.0	459.2
509	2374.6	35950	2739.1	3256.5	2901.1	2442.2	519.8	608.1	675.6	466.6
510	2264.7	70224	2596.2	3096.8	2793.8	2311.8	512.4	600.2	660.0	459.2
511	2374.6	75950	2739.1	3256.5	2901.1	2442.2	519.8	608.1	675.6	466.6
512	1178.2	25157	1351.9	1610.8	1421.7	1214.3	279.2	339.8	377.2	230.0
513	3359.6	86266	718.2	812.2	811.2	665.9	416.8	505.8	404.7	346.7
514	446.5	7144	507.1	616.1	528.2	471.1	117.2	153.6	196.3	95.4
515	32276.1	4712052	43171.3	48413.2	44202.8	40187.1	2217.9	2401.9	3189.2	2070.8
516	31956.7	4617103	42688.4	47876.5	43701.9	39728.3	2217.9	2401.1	3181.3	2074.6

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
517	TUPSUP.C	B	1.2	694	315	433	474	89	41	250	2181	1720	3901	2211.1
518	TUPSUP.C	B	1.4	587	318	534	394	76	71	218	1911	1493	3404	1913.1
519	TUPSW.C	U	1.4	237	90	106	159	48	39	230	742	597	1339	2010.6
520	TUPSW.C	U	1.5	236	93	112	158	48	39	229	740	596	1336	2001.3
521	TUPSW.C	U	1.7	245	95	114	164	51	39	232	758	607	1365	2029.2
522	TUPSW.C	U	1.9	250	105	140	167	54	39	238	783	621	1404	2085.1
523	TURNON.C	U	1.0	13	5	19	4	3	14	13	32	21	53	101.4
524	TXTODISP.C	E	1.4	39	92	34	.	9	21	31	131	93	224	245.8
525	TXTODISP.C	E	1.6	73	39	55	.	24	23	51	218	146	364	393.3
526	TXTOSCRN.C	U	1.4	8	9	11	.	1	11	11	19	15	34	76.1
527	TXTST.C	U	1.4	212	138	104	103	49	34	110	465	335	800	918.9
528	TXTST.C	U	1.5	217	140	107	105	51	34	111	473	341	814	927.3
529	TXTST.C	U	1.9	232	155	148	114	50	37	125	512	362	874	1063.5
530	UNITCOMP.C	U	1.4	24	12	29	9	6	14	8	50	28	78	77.3
531	UNPACK.C	G	1.0	20	16	23	13	2	19	22	65	46	111	178.8
532	UPDATEW.C	Q	1.0	37	25	31	21	7	22	29	181	78	259	239.0
533	UPDONOF.C	Q	1.0	23	32	41	10	3	11	13	40	28	68	86.2
534	UPDSCRN.C	Q	1.0	169	32	50	70	41	36	47	572	356	928	447.2
535	UPDSREG.C	Q	1.0	76	32	43	41	11	26	37	329	198	527	315.0
536	UPDWLIN.C	Q	1.0	52	27	37	33	9	25	40	220	116	336	229.0
537	UPOBJ.C	Q	1.0	23	21	28	13	7	19	19	82	61	143	161.4
538	UPSCROLL.C	U	1.4	43	36	21	.	10	25	35	135	100	243	295.6
539	UPSCROLL.C	U	1.6	39	37	25	.	10	25	32	119	92	211	276.1
540	VDBNAME.C	U	1.4	17	9	7	.	1	9	17	52	36	88	98.0
541	VDBTOS.C	U	1.4	42	26	42	.	10	27	43	148	102	250	361.7
542	VDXDEL.C	F	1.4	82	37	38	.	9	23	54	234	225	459	414.8

OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
517	31929.2	4503294	42675.9	47863.1	43689.1	39718.7	2211.1	2391.4	3171.9	2069.9
518	27289.2	3831318	36572.3	41113.5	37443.2	34024.9	1913.1	2085.8	2763.7	1775.2
519	10807.7	547034	12580.5	13327.3	13108.0	11668.8	2010.6	2215.4	2540.2	1878.5
520	10776.3	546909	12547.8	13292.6	13073.8	11636.9	2001.3	2206.3	2530.9	1869.3
521	11032.7	562853	12863.1	13628.5	13392.4	11933.4	2029.2	2233.7	2566.1	1896.9
522	11391.7	579612	13288.8	14081.8	13795.4	12319.1	2085.1	2288.7	2618.0	1952.2
523	252.0	2850	252.2	332.5	275.5	212.7	101.4	134.6	133.7	77.3
524	1276.9	40222	1529.5	1795.2	1536.6	1316.2	245.8	303.6	336.5	197.7
525	2260.2	74411	2743.2	3276.6	2795.2	2378.6	393.3	467.2	543.8	332.8
526	151.6	1137	139.3	186.1	186.1	121.1	76.1	104.0	104.0	61.7
527	5735.9	296965	6930.4	8081.9	7394.3	6327.6	918.9	1041.1	1195.9	836.2
528	5844.4	305228	7072.0	8249.3	7242.8	6462.9	927.2	1049.7	1211.0	844.3
529	6415.0	343692	7684.9	8851.9	8177.2	7024.4	1063.5	1024.7	1370.2	973.5
530	347.8	8522	416.8	505.8	404.7	346.7	77.3	998.1	112.0	52.8
531	594.7	11813	645.5	795.5	733.9	534.6	178.8	226.5	246.1	138.6
532	1469.2	43467	1847.7	2123.7	1952.1	1682.1	239.0	296.4	341.9	193.8
533	311.8	3693	347.5	451.9	421.9	298.9	86.2	116.1	114.8	68.5
534	5916.0	806595	8256.8	9556.0	8408.4	7321.4	447.2	537.0	767.9	366.2
535	3150.0	219140	4261.7	4943.6	4494.4	3905.9	315.0	384.0	539.4	263.9
536	2023.5	73352	2507.4	2869.0	2659.0	2265.5	329.0	398.9	458.9	274.9
537	750.5	22889	883.1	1119.1	971.9	772.6	161.4	206.1	239.0	124.5
538	1435.4	55364	1684.9	2000.9	1680.5	1469.3	295.6	361.8	462.7	249.4
539	1230.7	44230	1420.7	1742.7	1435.5	1223.4	276.1	339.8	405.9	230.2
540	413.6	3942	482.5	592.2	592.2	447.3	98.0	128.4	128.4	82.1
541	1532.3	49070	1747.6	2001.2	1788.1	1532.4	361.7	444.6	473.9	296.9
542	2876.5	137830	3599.8	4477.8	4082.5	3239.1	414.8	508.2	678.1	360.3

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST
543	VDXINS.C	U	1.4	139	29	17	.	17	26	50	322	363	685	404.4
544	VDXWRITE.C	U	1.4	12	7	0	.	2	16	17	37	29	66	133.5
545	VSCROLL.C	Q	1.0	86	23	42	55	12	25	24	250	164	414	226.1
546	WAITGET.C	U	1.4	50	21	23	22	14	30	35	140	85	225	326.7
547	WHICHIS.C	U	1.4	13	4	0	.	2	16	13	35	28	63	112.1
548	WILLBUMP.C	U	1.4	15	18	28	.	7	14	18	38	30	68	128.4
549	WINOFF.C	Q	1.0	20	23	30	13	5	20	21	97	49	146	178.7
550	WINON.C	Q	1.0	15	20	28	9	4	15	15	43	28	71	117.2
551	WINTRST.C	Q	1.0	56	27	38	25	18	22	24	183	118	301	208.1
552	WOP.C	A	1.4	176	41	93	72	25	26	36	379	249	628	308.3
553	WOP.C	A	1.6	200	48	101	88	28	33	51	464	304	768	455.8
554	WORDER.C	G	1.0	69	37	50	37	15	23	46	199	167	366	358.1
555	WORDER.C	G	1.1	70	39	55	37	15	24	49	204	170	374	385.2
556	WORDS.C	H	1.4	60	25	21	22	34	21	32	155	120	275	252.2
557	WORKMSG.C	U	1.4	22	5	0	6	1	16	19	46	26	72	144.7
558	WRITEDR.C	U	1.4	28	20	29	.	3	22	30	100	58	158	245.3
559	WRITEV.C	U	1.4	32	27	35	.	4	22	35	84	72	156	277.6
560	WSEGOP.C	B	1.4	167	38	77	62	24	23	34	321	223	554	277.0
561	WSEGOP.C	B	1.6	191	42	85	78	27	30	49	406	278	684	422.3
562	WWDCCNT.C	G	1.4	34	14	21	16	5	23	15	79	58	137	220.1
563	WWHEEL.C	I	1.0	499	163	195	276	89	45	192	1593	1156	2749	1703.4
564	WWHEEL.C	I	1.1	527	163	216	301	99	47	207	1710	1245	2955	1853.6
565	WWHEEL.C	I	1.2	588	173	227	332	114	47	231	1835	1335	3170	2074.8
566	WWHEEL.C	I	1.3	410	141	190	226	79	42	187	1193	860	2053	1637.7
567	WWHEEL.C	I	1.4	411	142	193	226	79	42	188	1196	862	2058	1646.7
568	WWHEEL.C	I	1.5	411	143	196	226	79	43	189	1210	874	2084	1662.6

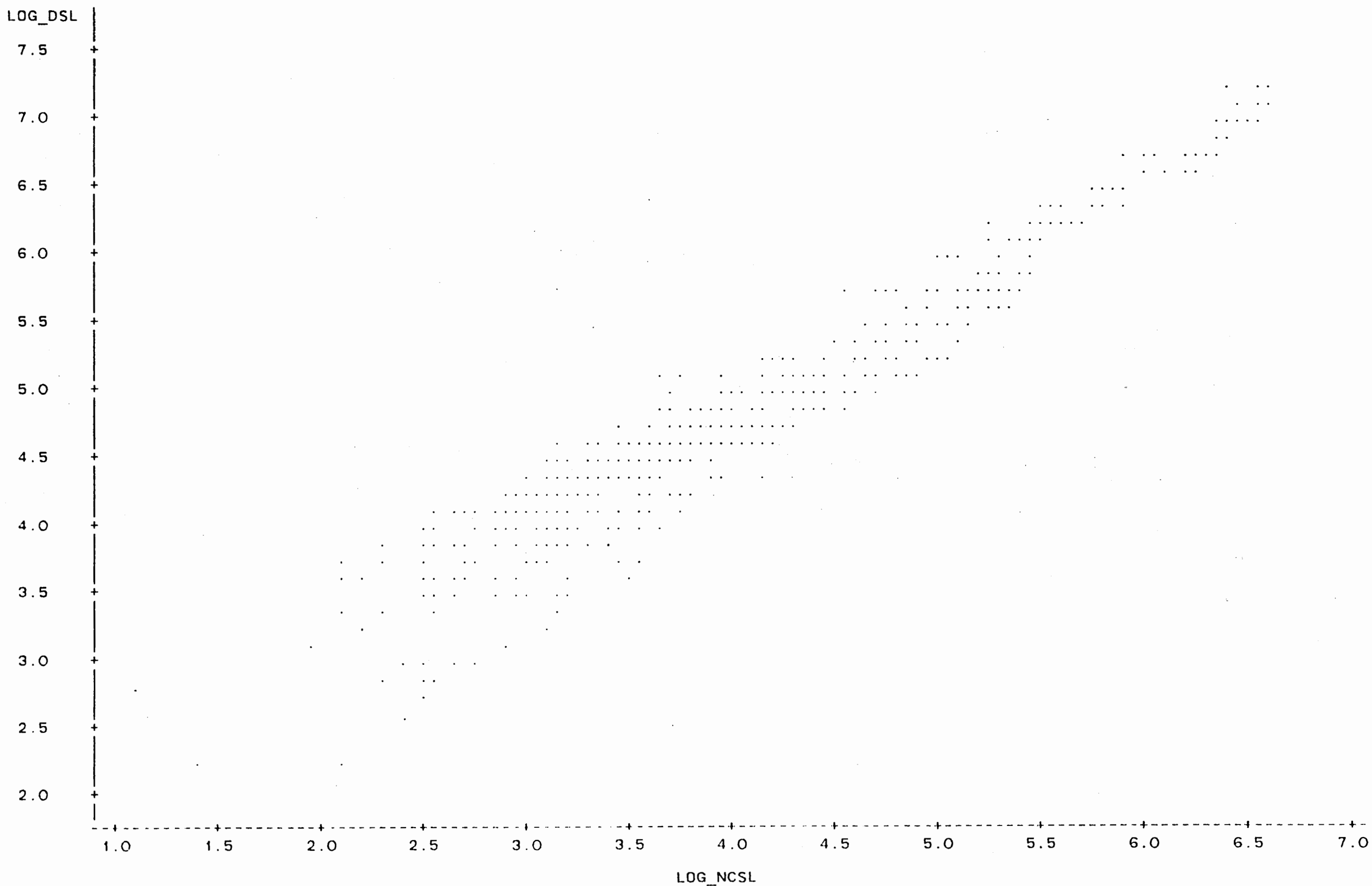
OBS	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
543	4279.8	403930	5769.4	7668.7	7048.1	5209.7	404.4	489.9	628.8	344.5
544	332.9	4544	333.6	370.5	381.4	273.5	133.5	184.0	175.5	99.3
545	2324.5	198550	3198.1	3761.7	3379.2	2853.1	226.1	282.2	321.3	177.1
546	1355.0	49362	1542.9	1869.6	1573.4	1341.7	326.7	398.9	451.5	271.3
547	306.1	5274	314.1	413.9	312.7	255.3	112.1	147.2	141.4	81.3
548	340.0	3967	346.6	429.0	359.7	273.8	128.4	166.5	150.8	96.6
549	782.2	18251	915.3	1084.3	980.2	819.6	178.7	226.5	248.0	138.3
550	348.4	4878	367.9	459.5	402.0	317.9	117.2	153.6	153.1	90.6
551	1662.6	89919	2187.5	2555.9	2269.2	1954.3	208.1	261.1	337.4	176.6
552	3739.2	336220	5228.6	6030.7	5436.0	4854.9	308.3	376.6	424.7	259.3
553	4909.3	482844	6617.5	7615.4	6921.3	6125.8	455.8	544.8	596.8	385.7
554	2235.7	93341	2752.8	3213.8	2878.8	2469.2	358.1	475.9	571.7	307.4
555	2315.0	96379	2824.8	3302.4	2960.4	2536.5	385.2	506.1	599.2	333.3
556	1575.2	62023	1956.6	2324.1	1923.2	1665.0	252.2	310.8	411.0	208.3
557	369.3	4043	376.3	664.4	664.4	320.3	144.7	186.1	186.1	118.5
558	900.7	19154	1004.1	1153.4	1121.6	869.8	245.3	325.0	321.3	207.1
559	909.9	20590	981.2	1233.6	1129.0	867.3	277.6	356.0	392.4	226.5
560	3173.1	239335	4412.4	5154.7	4595.1	4100.4	277.0	339.8	374.7	234.9
561	4311.8	366942	5775.2	6714.0	6053.1	5345.9	422.3	505.8	544.0	359.7
562	765.1	20414	837.8	1032.5	900.1	715.5	220.1	275.1	284.6	179.6
563	21686.2	2937796	28707.8	32502.1	29152.3	26446.3	1703.4	1879.0	2676.8	1549.4
564	23606.6	3336565	31166.0	35354.6	31623.6	28773.0	1853.6	2038.6	2928.1	1707.5
565	25737.0	3495391	33755.1	38282.5	34332.2	31200.1	2074.8	2266.6	3197.1	1925.9
566	16093.9	1554308	20576.4	23425.7	21227.7	18948.2	1637.7	1804.5	2317.9	1502.7
567	16146.0	1554656	20633.7	23500.6	21299.4	18997.4	1646.7	1813.8	2326.9	1511.5
568	16376.0	1628159	20931.6	23825.8	21610.3	19232.5	1662.6	1832.4	2354.2	1524.9

OBS	FILENAME	AUTH	VER	NCSL	BL	COM	SC	VGC	N1	N2	N1TOT	N2TOT	HN	HNEST	HV	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
569	YYPARSE.C	H	1.4	37	14	20	25	6	18	31	96	64	160	228.6	898.4	16692	1016.2	1206.6	1061.1	854.9	228.6	282.2	301.5	188.6

APPENDIX B

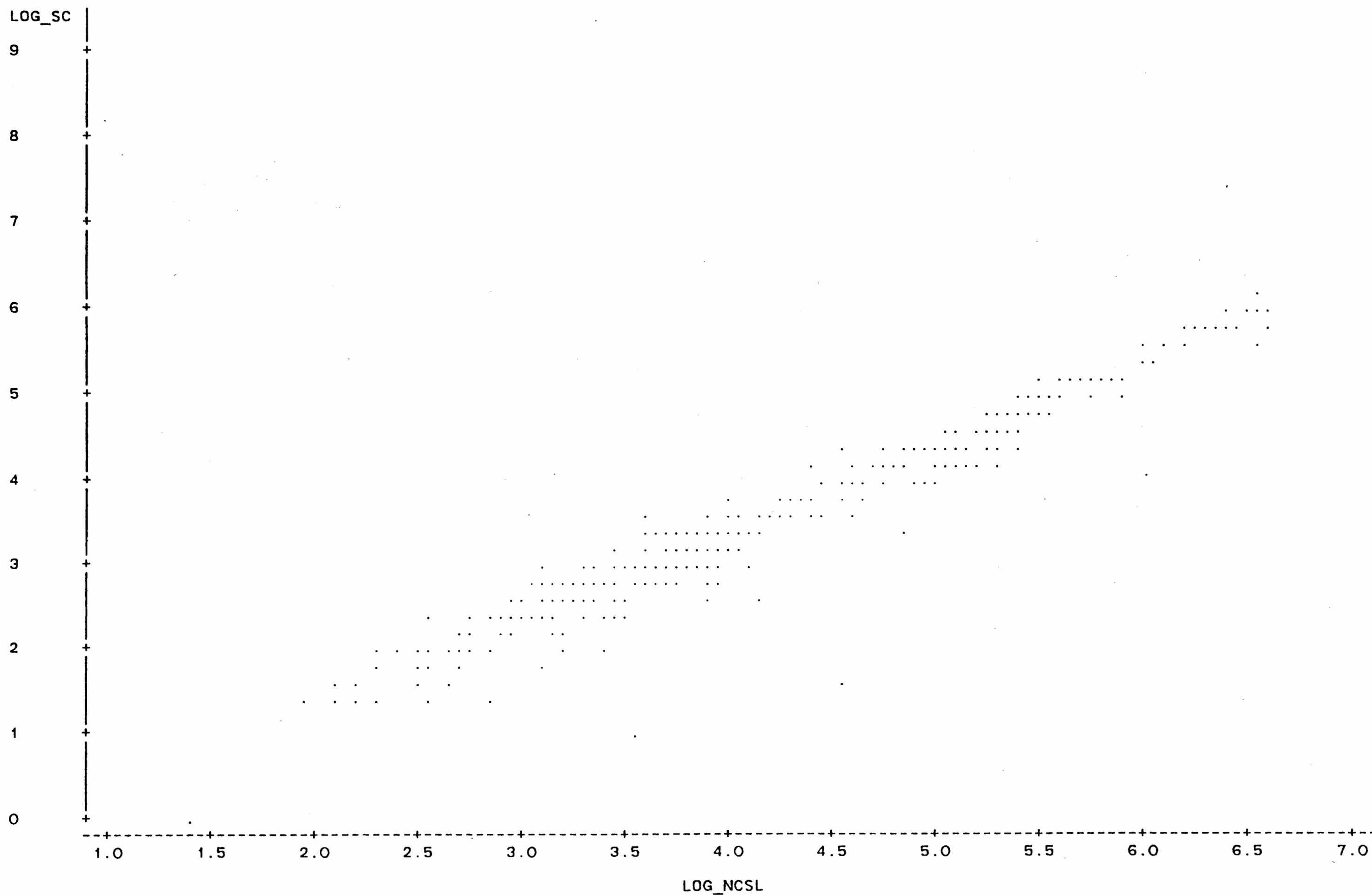
SAS FIGURES AND TABLES

PLOT OF LOG_DSL*LOG_NCSL SYMBOL USED IS .



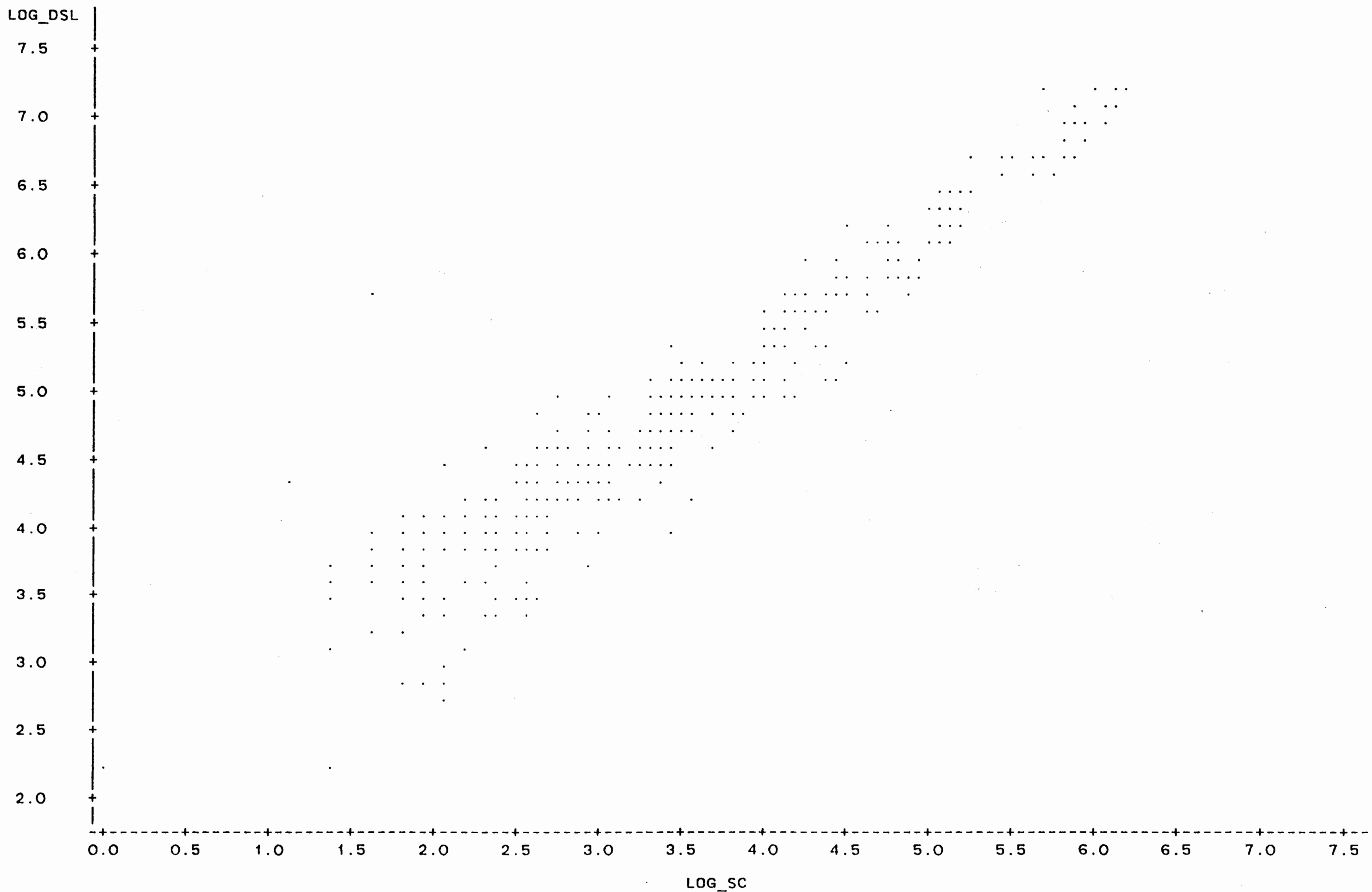
NOTE: 252 OBS HIDDEN

PLOT OF LOG_SC*LOG_NCSL SYMBOL USED IS .



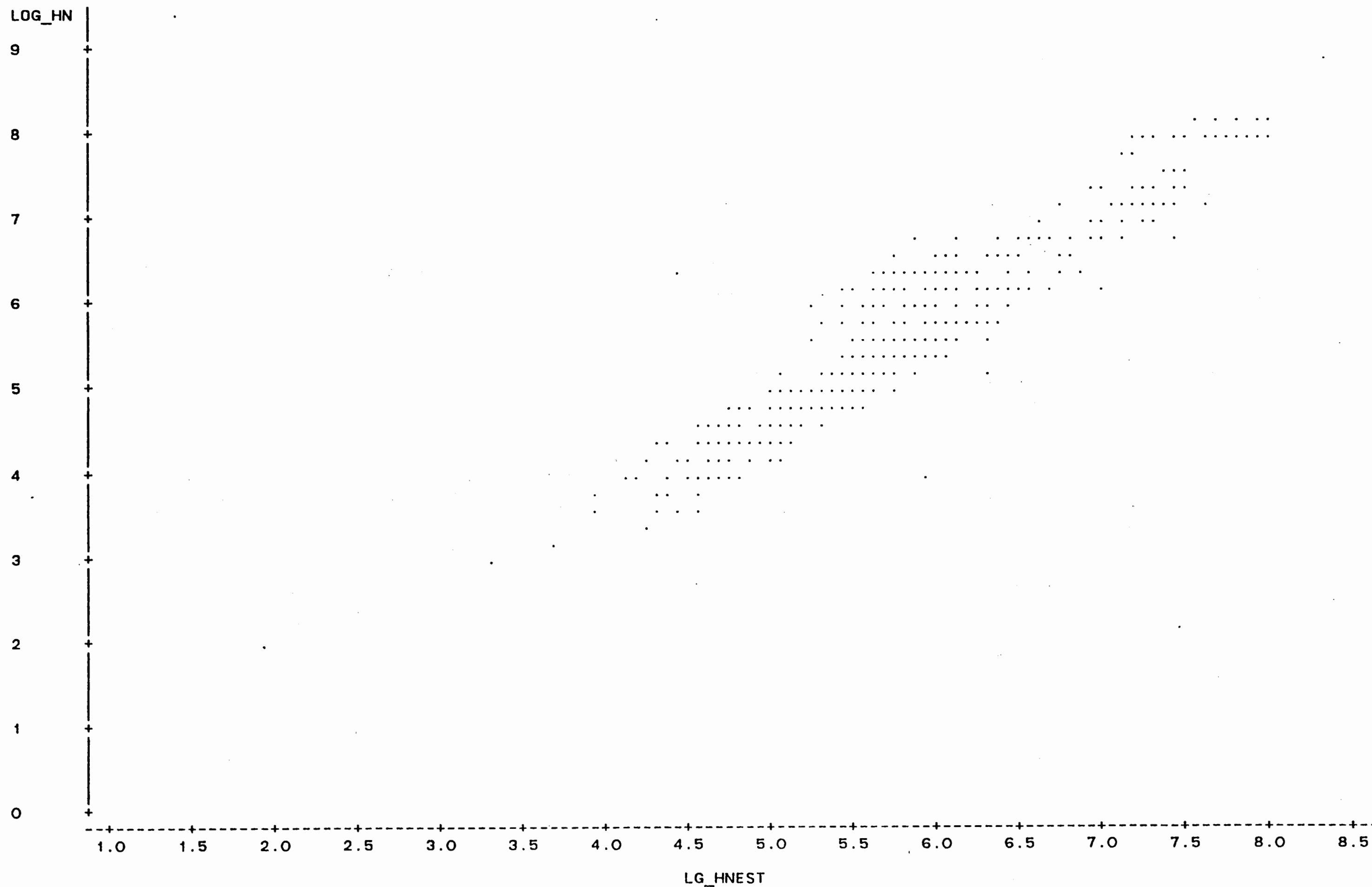
NOTE: 99 OBS HAD MISSING VALUES 260 OBS HIDDEN

PLOT OF LOG_DSL*LOG_SC SYMBOL USED IS .



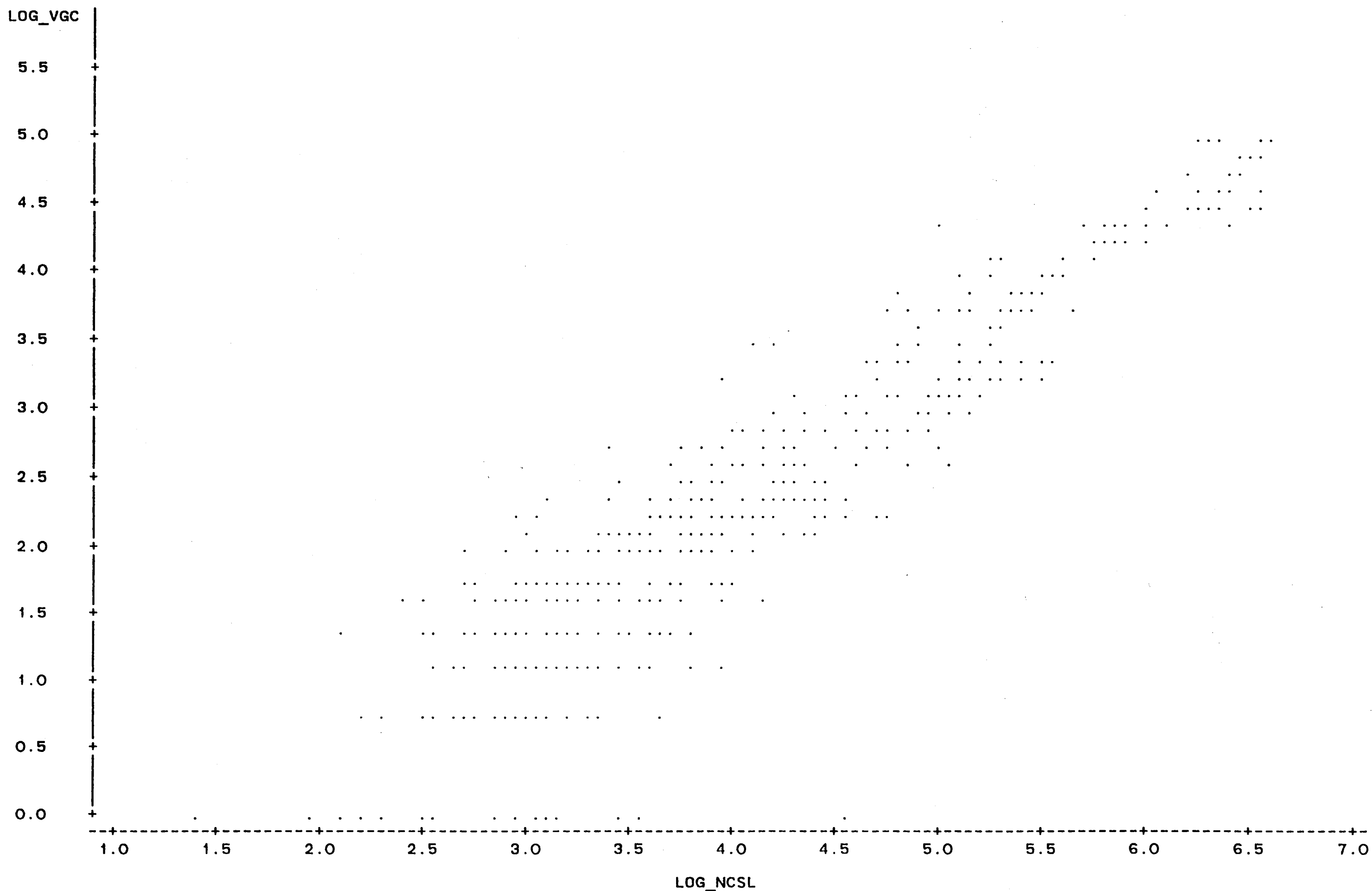
NOTE: 99 OBS HAD MISSING VALUES 209 OBS HIDDEN

PLOT OF LOG_HN*LG_HNEST SYMBOL USED IS .



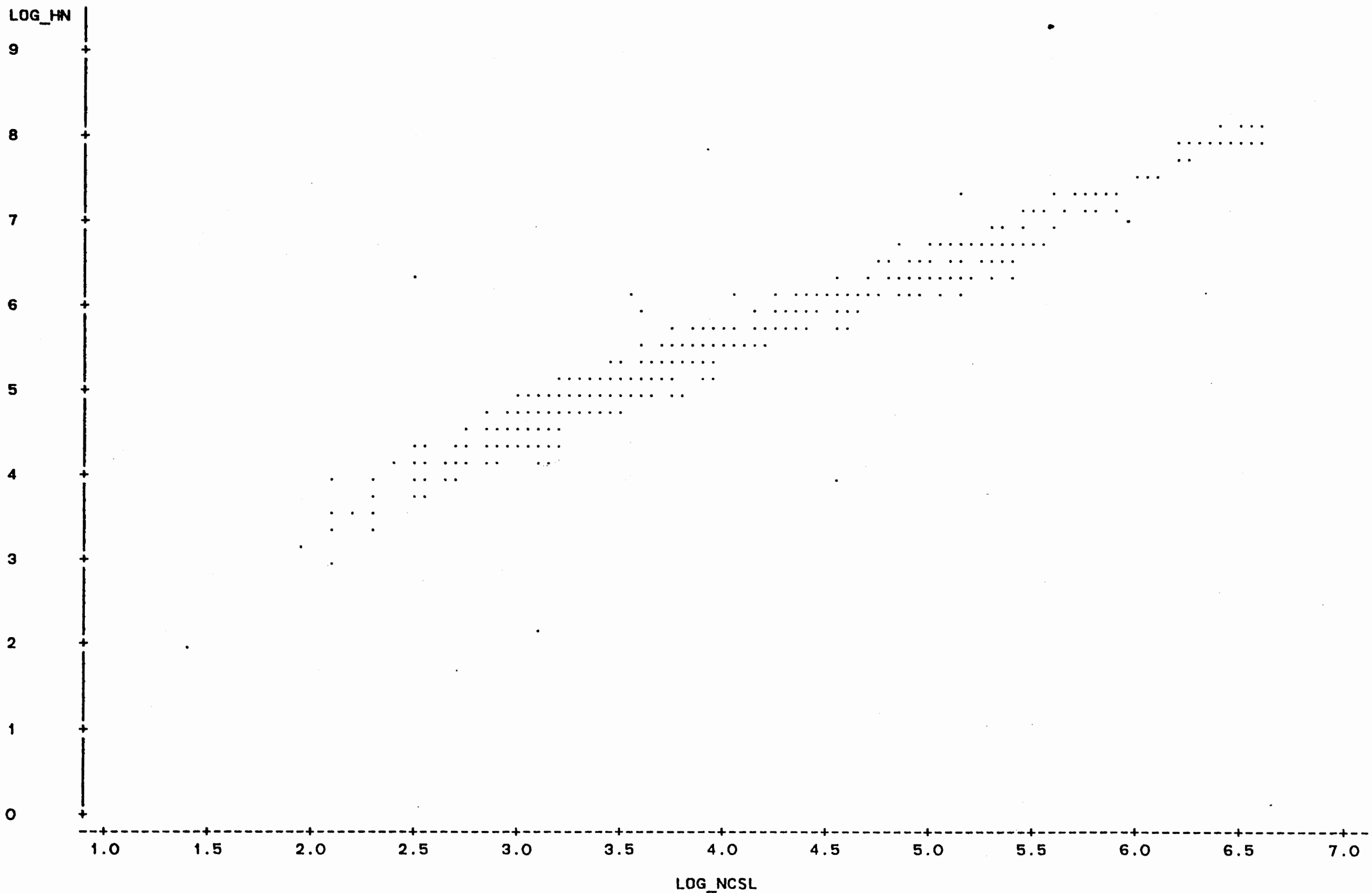
NOTE: 2 OBS HAD MISSING VALUES 326 OBS HIDDEN

PLOT OF LOG_VGC*LOG_NCSL SYMBOL USED IS .



NOTE: 241 OBS HIDDEN

PLOT OF LOG_HN*LOG_NCSL SYMBOL USED IS .



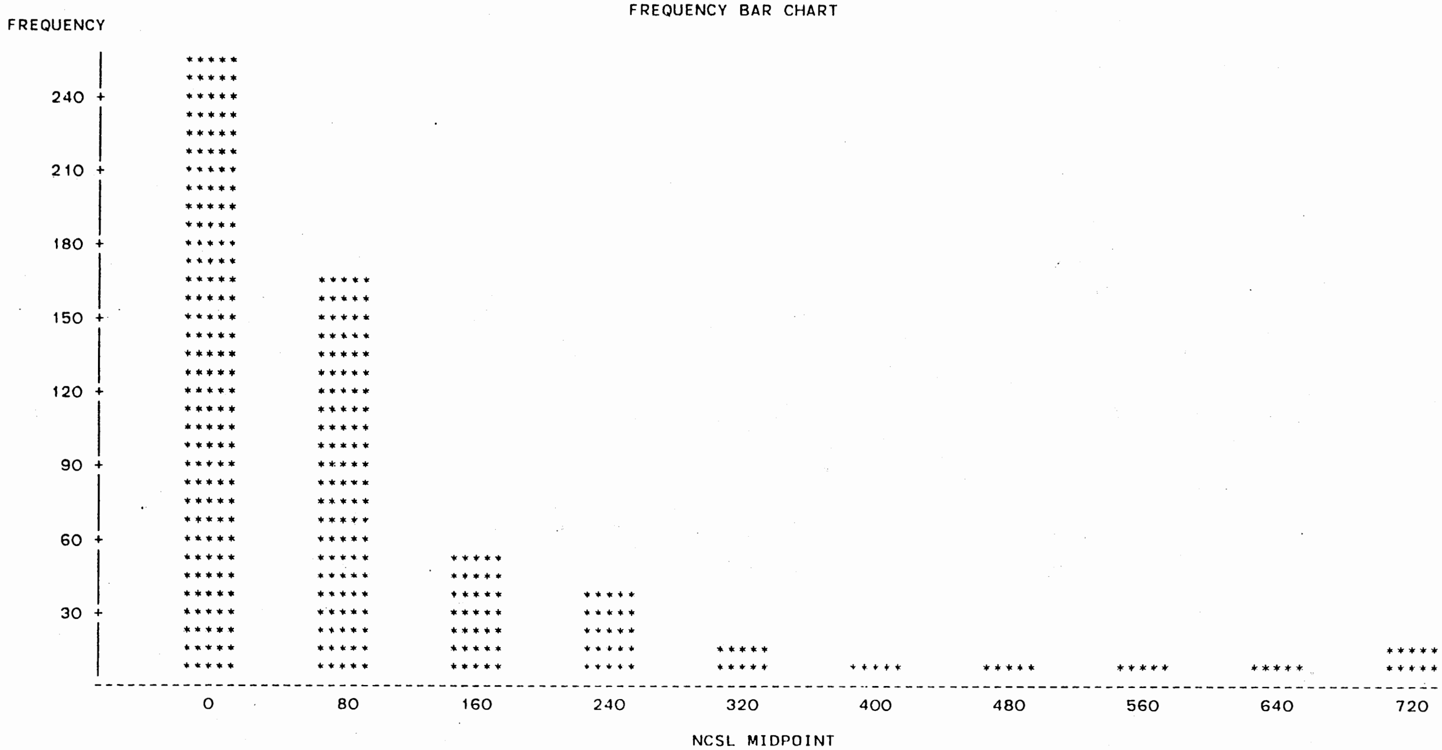
NOTE: 339 OBS HIDDEN

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
LOG_NCSL	569	4.03975304	1.13538731	2298.61948000	1.38629436	6.59167373
LOG_DSL	569	4.84804624	1.01446725	2758.53830951	2.19722458	7.27655640
LOG_SC	470	3.50457372	1.23420694	1647.14964721	0.00000000	6.16541785

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	LOG_NCSL	LOG_DSL	LOG_SC
LOG_NCSL	1.00000	0.96835	0.98045
	0.0000	0.0001	0.0001
	569	569	470
LOG_DSL	0.96835	1.00000	0.95513
	0.0001	0.0000	0.0001
	569	569	470
LOG_SC	0.98045	0.95513	1.00000
	0.0001	0.0001	0.0000
	470	470	470

FREQUENCY BAR CHART



VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
PC	25	3.4400000	1.6350331	86.0000	1.000000000E+00	6.000000
NCSL	569	114.3181019	159.7692076	65047.0000	4.00000000	729.000000
VGC	569	22.0000000	31.0892627	12518.0000	1.000000000E+00	154.000000
HE	568	358546.2339437	865006.6375821	203654260.8800	16.30000000	4930184.300000
R1	569	5056.8270650	8479.9304134	2877334.6000	12.80000000	43171.300000
R2	569	5826.7532513	9615.2736555	3315422.6000	3.80000000	48762.300000
R3	569	5321.4179262	8809.6169786	3027886.8000	24.00000000	45756.600000
R4	567	4638.1223986	7876.9777362	2629815.4000	12.80000000	40187.100000
R1U	569	529.8087873	600.6173424	301461.2000	6.80000000	3074.300000
R2U	569	612.6987698	644.9549871	348625.6000	11.60000000	3306.700000
R3U	569	722.2734622	808.0965663	410973.6000	11.60000000	3895.600000
R4U	569	477.4919156	571.3509906	271692.9000	6.80000000	2912.400000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=O / NUMBER OF OBSERVATIONS

	PC	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
PC	1.00000	0.64789	0.67164	0.56892	0.59845	0.59684	0.59108	0.59679	0.52267	0.51880	0.56550	0.52417
	0.0000	0.0005	0.0002	0.0030	0.0016	0.0017	0.0019	0.0017	0.0074	0.0079	0.0032	0.0072
	25	25	25	25	25	25	25	25	25	25	25	25
NCSL	0.64789	1.00000	0.96199	0.92697	0.98113	0.98200	0.98202	0.98261	0.93890	0.93564	0.95706	0.93806
	0.0005	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	25	569	569	568	569	569	569	567	569	569	569	569
VGC	0.67164	0.96199	1.00000	0.86535	0.93190	0.93264	0.93039	0.93190	0.89765	0.89546	0.91767	0.89662
	0.0002	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	25	569	569	568	569	569	569	567	569	569	569	569
HE	0.56892	0.92697	0.86535	1.00000	0.96705	0.96674	0.96628	0.96780	0.82662	0.82142	0.85778	0.82615
	0.0030	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	25	568	568	568	568	568	568	566	568	568	568	568
R1	0.59845	0.98113	0.93190	0.96705	1.00000	0.99950	0.99818	0.99854	0.92770	0.92366	0.94758	0.92547
	0.0016	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	25	569	569	568	569	569	569	567	569	569	569	569
R2	0.59684	0.98200	0.93264	0.96674	0.99950	1.00000	0.99851	0.99854	0.92786	0.92366	0.94745	0.92564
	0.0017	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	25	569	569	568	569	569	569	567	569	569	569	569
R3	0.59108	0.98202	0.93039	0.96628	0.99818	0.99851	1.00000	0.99965	0.93212	0.92821	0.95070	0.92980
	0.0019	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001
	25	569	569	568	569	569	569	567	569	569	569	569
R4	0.59679	0.98261	0.93190	0.96780	0.99854	0.99854	0.99965	1.00000	0.92941	0.92551	0.94934	0.92709
	0.0017	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001
	25	567	567	566	567	567	567	567	567	567	567	567
R1U	0.52267	0.93890	0.89765	0.82662	0.92770	0.92786	0.93212	0.92941	1.00000	0.99758	0.99354	0.99606
	0.0074	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001
	25	569	569	568	569	569	569	567	569	569	569	569
R2U	0.51880	0.93564	0.89546	0.82142	0.92366	0.92366	0.92821	0.92551	0.99758	1.00000	0.99241	0.99341
	0.0079	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001
	25	569	569	568	569	569	569	567	569	569	569	569
R3U	0.56550	0.95706	0.91767	0.85778	0.94758	0.94745	0.95070	0.94934	0.99354	0.99241	1.00000	0.98984
	0.0032	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001
	25	569	569	568	569	569	569	567	569	569	569	569
R4U	0.52417	0.93806	0.89662	0.82615	0.92547	0.92564	0.92980	0.92709	0.99606	0.99341	0.98984	1.00000
	0.0072	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000
	25	569	569	568	569	569	569	567	569	569	569	569

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
ERR	569	2.847100176E-01	8.618243191E-01	162.0000	0.00000000	8.000000
CF	90	1.8000000	1.4078432	162.0000	1.000000000E+00	8.000000
NCSL	569	114.3181019	159.7692076	65047.0000	4.00000000	729.000000
VGC	569	22.0000000	31.0892627	12518.0000	1.000000000E+00	154.000000
HE	568	358546.2339437	865006.6375821	203654260.8800	16.30000000	4930184.300000
R1	569	5056.8270650	8479.9304134	2877334.6000	12.80000000	43171.300000
R2	569	5826.7532513	9615.2736555	3315422.6000	3.80000000	48762.300000
R3	569	5321.4179262	8809.6169786	3027886.8000	24.00000000	45756.600000
R4	567	4638.1223986	7876.9777362	2629815.4000	12.80000000	40187.100000
R1U	569	529.8087873	600.6173424	301461.2000	6.80000000	3074.300000
R2U	569	612.6987698	644.9549871	348625.6000	11.60000000	3306.700000
R3U	569	722.2734622	808.0965663	410973.6000	11.60000000	3895.600000
R4U	569	477.4919156	571.3509906	271692.9000	6.80000000	2912.400000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	ERR	CF	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
ERR	1.00000	1.00000	0.29410	0.28708	0.22342	0.27996	0.28034	0.27766	0.27725	0.33409	0.33015	0.32511	0.32924
	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	569	90	569	569	568	569	569	569	567	569	569	569	569
CF	1.00000	1.00000	0.32323	0.38371	0.25943	0.30327	0.30679	0.30331	0.30199	0.35034	0.34209	0.34125	0.34680
	0.0001	0.0000	0.0019	0.0002	0.0135	0.0037	0.0033	0.0037	0.0040	0.0007	0.0010	0.0010	0.0008
	90	90	90	90	90	90	90	90	89	90	90	90	90
NCSL	0.29410	0.32323	1.00000	0.96199	0.92697	0.98113	0.98200	0.98202	0.98261	0.93890	0.93564	0.95706	0.93806
	0.0001	0.0019	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	569	90	569	569	568	569	569	569	567	569	569	569	569
VGC	0.28708	0.38371	0.96199	1.00000	0.86535	0.93190	0.93264	0.93039	0.93190	0.89765	0.89546	0.91767	0.89662
	0.0001	0.0002	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	569	90	569	569	568	569	569	569	567	569	569	569	569
HE	0.22342	0.25943	0.92697	0.86535	1.00000	0.96705	0.96674	0.96628	0.96780	0.82662	0.82142	0.85778	0.82615
	0.0001	0.0135	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	568	90	568	568	568	568	568	568	566	568	568	568	568
R1	0.27996	0.30327	0.98113	0.93190	0.96705	1.00000	0.99950	0.99818	0.99854	0.92770	0.92366	0.94758	0.92547
	0.0001	0.0037	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	569	90	569	569	568	569	569	569	567	569	569	569	569
R2	0.28034	0.30679	0.98200	0.93264	0.96674	0.99950	1.00000	0.99851	0.99854	0.92786	0.92366	0.94745	0.92564
	0.0001	0.0033	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	569	90	569	569	568	569	569	569	567	569	569	569	569
R3	0.27766	0.30331	0.98202	0.93039	0.96628	0.99818	0.99851	1.00000	0.99965	0.93212	0.92821	0.95070	0.92980
	0.0001	0.0037	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001
	569	90	569	569	568	569	569	569	567	569	569	569	569
R4	0.27725	0.30199	0.98261	0.93190	0.96780	0.99854	0.99854	0.99965	1.00000	0.92941	0.92551	0.94934	0.92709
	0.0001	0.0040	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001
	567	89	567	567	566	567	567	567	567	567	567	567	567
R1U	0.33409	0.35034	0.93890	0.89765	0.82662	0.92770	0.92786	0.93212	0.92941	1.00000	0.99758	0.99354	0.99606
	0.0001	0.0007	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001
	569	90	569	569	568	569	569	569	567	569	569	569	569
R2U	0.33015	0.34209	0.93564	0.89546	0.82142	0.92366	0.92366	0.92821	0.92551	0.99758	1.00000	0.99241	0.99341
	0.0001	0.0010	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001
	569	90	569	569	568	569	569	569	567	569	569	569	569
R3U	0.32511	0.34125	0.95706	0.91767	0.85778	0.94758	0.94745	0.95070	0.94934	0.99354	0.99241	1.00000	0.98984
	0.0001	0.0010	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001
	569	90	569	569	568	569	569	569	567	569	569	569	569
R4U	0.32924	0.34680	0.93806	0.89662	0.82615	0.92547	0.92564	0.92980	0.92709	0.99606	0.99341	0.98984	1.00000
	0.0001	0.0008	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000
	569	90	569	569	568	569	569	569	567	569	569	569	569

GENERAL LINEAR MODELS PROCEDURE

DEPENDENT VARIABLE INFORMATION

NUMBER OF OBSERVATIONS IN DATA SET = 569

NOTE: ALL DEPENDENT VARIABLES ARE CONSISTENT WITH RESPECT TO THE PRESENCE OR ABSENCE OF MISSING VALUES. HOWEVER, ONLY 89 OBSERVATIONS CAN BE USED IN THIS ANALYSIS.

GENERAL LINEAR MODELS PROCEDURE

DEPENDENT VARIABLE: CF

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PR > F	R-SQUARE	C.V.
MODEL	8	48.01439889	6.00179986	3.74	0.0009	0.272253	70.4555
ERROR	80	128.34515167	1.60431440		ROOT MSE		CF MEAN
CORRECTED TOTAL	88	176.35955056			1.26661533		1.79775281

SOURCE	DF	TYPE I SS	F VALUE	PR > F	DF	TYPE III SS	F VALUE	PR > F
LOG_HN	1	17.74214811	11.06	0.0013	1	1.18626493	0.74	0.3924
LOG_NCSL	1	0.44650944	0.28	0.5993	1	2.87740865	1.79	0.1843
LOG_VGC	1	2.88962572	1.80	0.1834	1	0.44724118	0.28	0.5990
R1U	1	3.67054538	2.29	0.1343	1	12.23936502	7.63	0.0071
R2U	1	8.57832485	5.35	0.0233	1	8.97029706	5.59	0.0205
R4U	1	5.94734352	3.71	0.0577	1	5.19571733	3.24	0.0757
R4	1	0.37948663	0.24	0.6280	1	8.54936828	5.33	0.0236
R2	1	8.36041525	5.21	0.0251	1	8.36041525	5.21	0.0251

PARAMETER	ESTIMATE	T FOR HO: PARAMETER=0	PR > |T|	STD ERROR OF ESTIMATE
INTERCEPT	-0.35332009	-0.12	0.9066	3.00049396
LOG_HN	0.96542967	0.86	0.3924	1.12272795
LOG_NCSL	-1.20614811	-1.34	0.1843	0.90062627
LOG_VGC	0.25685438	0.53	0.5990	0.48647533
R1U	0.03081291	2.76	0.0071	0.01115573
R2U	-0.01115693	-2.36	0.0205	0.00471831
R4U	-0.01885666	-1.80	0.0757	0.01047820
R4	-0.00130429	-2.31	0.0236	0.00056500
R2	0.00104961	2.28	0.0251	0.00045979

VER=1

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
NCSL	156	72.2756410	99.7287013	11275.00000	8.00000000	698.000000
VGC	156	14.6602564	19.7919817	2287.00000	1.00000000	122.000000
HE	156	198509.4217949	593029.2688245	30967469.80000	317.10000000	4712052.200000
R1	156	3078.3160256	5576.8953422	480217.30000	76.20000000	43171.300000
R2	156	3592.8839744	6282.1788458	560489.90000	104.00000000	48413.200000
R3	156	3308.0250000	5710.6797503	516051.90000	104.00000000	44202.800000
R4	155	2842.9877419	5168.5971773	440663.10000	71.30000000	40187.100000
R1U	156	344.1166667	321.9998659	53682.20000	27.10000000	2217.900000
R2U	156	412.5544872	354.0372789	64358.50000	38.10000000	2401.900000
R3U	156	486.5974359	473.0041707	75909.20000	38.10000000	3189.200000
R4U	156	296.5923077	300.5569090	46268.40000	23.50000000	2070.800000

VER=1

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
NCSL	1.00000	0.92877	0.92545	0.96808	0.97087	0.98334	0.98401	0.92567	0.91927	0.94832	0.92419
	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	156	156	156	156	156	156	155	156	156	156	156
VGC	0.92877	1.00000	0.81408	0.87837	0.88381	0.89075	0.88857	0.83044	0.82600	0.85410	0.82486
	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	156	156	156	156	156	156	155	156	156	156	156
HE	0.92545	0.81408	1.00000	0.96935	0.96733	0.96498	0.96803	0.80115	0.78900	0.84029	0.80257
	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	156	156	156	156	156	156	155	156	156	156	156
R1	0.96808	0.87837	0.96935	1.00000	0.99883	0.98922	0.98941	0.88939	0.87939	0.91373	0.88962
	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	156	156	156	156	156	156	155	156	156	156	156
R2	0.97087	0.88381	0.96733	0.99883	1.00000	0.98941	0.98949	0.89191	0.88152	0.91593	0.89188
	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	156	156	156	156	156	156	155	156	156	156	156
R3	0.98334	0.89075	0.96498	0.98922	0.98941	1.00000	0.99982	0.91430	0.90658	0.93950	0.91413
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001
	156	156	156	156	156	156	155	156	156	156	156
R4	0.98401	0.88857	0.96803	0.98941	0.98949	0.99982	1.00000	0.91618	0.90780	0.93992	0.91576
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001
	155	155	155	155	155	155	155	155	155	155	155
R1U	0.92567	0.83044	0.80115	0.88939	0.89191	0.91430	0.91618	1.00000	0.99819	0.98940	0.99927
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001
	156	156	156	156	156	156	155	156	156	156	156
R2U	0.91927	0.82600	0.78900	0.87939	0.88152	0.90658	0.90780	0.99819	1.00000	0.98895	0.99687
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001
	156	156	156	156	156	156	155	156	156	156	156
R3U	0.94832	0.85410	0.84029	0.91373	0.91593	0.93950	0.93992	0.98940	0.98895	1.00000	0.98772
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001
	156	156	156	156	156	156	155	156	156	156	156
R4U	0.92419	0.82486	0.80257	0.88962	0.89188	0.91413	0.91576	0.99927	0.99687	0.98772	1.00000
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000
	156	156	156	156	156	156	155	156	156	156	156

VER=1.1

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
NCSL	21	224.952381	232.415248	4724.00000	13.00000000	694.000000
VGC	21	43.952381	46.574109	923.00000	2.00000000	144.000000
HE	21	1048779.133333	1507613.635022	22024361.80000	1489.10000000	4617103.400000
R1	21	11596.952381	13500.038288	243536.00000	222.30000000	42688.400000
R2	21	13170.009524	15186.636857	276570.20000	296.40000000	47876.500000
R3	21	11959.419048	13828.214751	251147.80000	272.70000000	43701.900000
R4	20	10805.975000	12852.040015	216119.50000	189.70000000	39728.300000
R1U	21	885.085714	752.129915	18586.80000	96.70000000	2494.600000
R2U	21	1004.623810	795.625901	21097.10000	128.40000000	2692.300000
R3U	21	1262.514286	1054.549685	26512.80000	124.10000000	3233.800000
R4U	21	805.014286	707.678511	16905.30000	82.10000000	2359.000000

VER=1.1

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
NCSL	1.00000	0.92928	0.94391	0.98906	0.98993	0.99027	0.99151	0.95351	0.95132	0.96923	0.95520
	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	21	21	21	21	21	21	20	21	21	21	21
VGC	0.92928	1.00000	0.83609	0.89751	0.89958	0.89987	0.89589	0.88475	0.88308	0.88544	0.88265
	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	21	21	21	21	21	21	20	21	21	21	21
HE	0.94391	0.83609	1.00000	0.97454	0.97251	0.97230	0.97501	0.81798	0.81692	0.87176	0.82144
	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	21	21	21	21	21	21	20	21	21	21	21
R1	0.98906	0.89751	0.97454	1.00000	0.99993	0.99991	0.99998	0.92237	0.92085	0.95095	0.92442
	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	21	21	21	21	21	21	20	21	21	21	21
R2	0.98993	0.89958	0.97251	0.99993	1.00000	0.99988	0.99990	0.92561	0.92388	0.95337	0.92753
	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	21	21	21	21	21	21	20	21	21	21	21
R3	0.99027	0.89987	0.97230	0.99991	0.99988	1.00000	0.99991	0.92568	0.92422	0.95264	0.92781
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001
	21	21	21	21	21	21	20	21	21	21	21
R4	0.99151	0.89589	0.97501	0.99998	0.99990	0.99991	1.00000	0.93291	0.93178	0.95664	0.93450
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001
	20	20	20	20	20	20	20	20	20	20	20
R1U	0.95351	0.88475	0.81798	0.92237	0.92561	0.92568	0.93291	1.00000	0.99861	0.98805	0.99960
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001
	21	21	21	21	21	21	20	21	21	21	21
R2U	0.95132	0.88308	0.81692	0.92085	0.92388	0.92422	0.93178	0.99861	1.00000	0.98959	0.99836
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001
	21	21	21	21	21	21	20	21	21	21	21
R3U	0.96923	0.88544	0.87176	0.95095	0.95337	0.95264	0.95664	0.98805	0.98959	1.00000	0.98832
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001
	21	21	21	21	21	21	20	21	21	21	21
R4U	0.95520	0.88265	0.82144	0.92442	0.92753	0.92781	0.93450	0.99960	0.99836	0.98832	1.00000
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000
	21	21	21	21	21	21	20	21	21	21	21

VER=1.11

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
NCSL	3	330.000000	295.653513	990.000000	40.00000000	631.000000
VGC	3	67.000000	59.101607	201.000000	6.00000000	124.000000
HE	3	1017227.966667	1333530.963239	3051683.900000	26281.80000000	2533398.900000
R1	3	14873.133333	15023.707110	44619.400000	1476.00000000	31116.300000
R2	3	16982.300000	16958.842700	50946.900000	1785.90000000	35276.500000
R3	3	15593.866667	15570.005052	46781.600000	1669.20000000	32405.200000
R4	3	13769.333333	14010.205129	41308.000000	1291.70000000	28925.400000
R1U	3	1543.600000	1051.134906	4630.800000	426.90000000	2513.800000
R2U	3	1686.466667	1111.212105	5059.400000	505.80000000	2711.900000
R3U	3	1953.500000	1383.019523	5860.500000	510.50000000	3267.500000
R4U	3	1452.500000	1010.380107	4357.500000	374.50000000	2377.900000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 3

	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
NCSL	1.00000	0.99587	0.95053	0.99122	0.99199	0.99169	0.99102	0.98828	0.98824	0.99361	0.98668
	0.0000	0.0579	0.2011	0.0844	0.0806	0.0821	0.0854	0.0975	0.0977	0.0720	0.1040
VGC	0.99587	1.00000	0.91842	0.97514	0.97644	0.97592	0.97480	0.99806	0.99804	0.99975	0.99737
	0.0579	0.0000	0.2589	0.1423	0.1385	0.1400	0.1432	0.0397	0.0399	0.0141	0.0462
HE	0.95053	0.91842	1.00000	0.98325	0.98215	0.98260	0.98353	0.89199	0.89186	0.90940	0.88734
	0.2011	0.2589	0.0000	0.1167	0.1205	0.1189	0.1157	0.2986	0.2988	0.2731	0.3051
R1	0.99122	0.97514	0.98325	1.00000	0.99998	0.99999	1.00000	0.95944	0.95936	0.96998	0.95652
	0.0844	0.1423	0.1167	0.0000	0.0038	0.0023	0.0010	0.1819	0.1821	0.1564	0.1884
R2	0.99199	0.97644	0.98215	0.99998	1.00000	1.00000	0.99997	0.96110	0.96102	0.97140	0.95823
	0.0806	0.1385	0.1205	0.0038	0.0000	0.0015	0.0048	0.1782	0.1783	0.1526	0.1846
R3	0.99169	0.97592	0.98260	0.99999	1.00000	1.00000	0.99999	0.96043	0.96035	0.97083	0.95755
	0.0821	0.1400	0.1189	0.0023	0.0015	0.0000	0.0032	0.1797	0.1799	0.1541	0.1862
R4	0.99102	0.97480	0.98353	1.00000	0.99997	0.99999	1.00000	0.95901	0.95892	0.96960	0.95607
	0.0854	0.1432	0.1157	0.0010	0.0048	0.0032	0.0000	0.1829	0.1831	0.1574	0.1894
R1U	0.98828	0.99806	0.89199	0.95944	0.96110	0.96043	0.95901	1.00000	1.00000	0.99919	0.99995
	0.0975	0.0397	0.2986	0.1819	0.1782	0.1797	0.1829	0.0000	0.0002	0.0255	0.0065
R2U	0.98824	0.99804	0.89186	0.95936	0.96102	0.96035	0.95892	1.00000	1.00000	0.99918	0.99995
	0.0977	0.0399	0.2988	0.1821	0.1783	0.1799	0.1831	0.0002	0.0000	0.0257	0.0063
R3U	0.99361	0.99975	0.90940	0.96998	0.97140	0.97083	0.96960	0.99919	0.99918	1.00000	0.99873
	0.0720	0.0141	0.2731	0.1564	0.1526	0.1541	0.1574	0.0255	0.0257	0.0000	0.0320
R4U	0.98668	0.99737	0.88734	0.95652	0.95823	0.95755	0.95607	0.99995	0.99995	0.99873	1.00000
	0.1040	0.0462	0.3051	0.1884	0.1846	0.1862	0.1894	0.0065	0.0063	0.0320	0.0000

VER=1.12

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
NCSL	2	681.000000	63.6396103	1362.000000	636.000000	726.000000
VGC	2	135.500000	16.2634560	271.000000	124.000000	147.000000
HE	2	3009205.950000	680955.3561232	6018411.900000	2527697.800000	3490714.100000
R1	2	34594.750000	4732.3121331	69189.500000	31248.500000	37941.000000
R2	2	39346.000000	5534.2419336	78692.000000	35432.700000	43259.300000
R3	2	35984.450000	4861.4298313	71968.900000	32546.900000	39422.000000
R4	2	31971.400000	4139.5445184	63942.800000	29044.300000	34898.500000
R1U	2	2643.650000	143.0477018	5287.300000	2542.500000	2744.800000
R2U	2	2844.650000	146.1589717	5689.300000	2741.300000	2948.000000
R3U	2	3518.050000	317.8444981	7036.100000	3293.300000	3742.800000
R4U	2	2506.200000	140.9970922	5012.400000	2406.500000	2605.900000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 2

	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
NCSL	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
VGC	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
HE	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
R1	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
R2	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
R3	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
R4	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
R1U	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
R2U	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
R3U	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
R4U	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000

VER=1.2

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
NCSL	17	220.2941176	207.696523	3745.00000	18.00000000	694.000000
VGC	17	41.3529412	43.471170	703.00000	2.00000000	154.000000
HE	17	887083.2000000	1414722.003981	15080414.40000	6376.20000000	4503294.200000
R1	17	10732.3176471	12461.065237	182449.40000	436.70000000	42675.900000
R2	17	12244.1647059	13966.274057	208150.80000	552.70000000	47863.100000
R3	17	11063.4705882	12710.770946	188079.00000	497.50000000	43689.100000
R4	17	9846.8941176	11576.445553	167397.20000	377.20000000	39718.700000
R1U	17	885.4764706	628.429438	15053.10000	156.20000000	2211.100000
R2U	17	1002.4941176	674.431290	17042.40000	199.40000000	2391.400000
R3U	17	1251.8882353	925.297925	21282.10000	212.70000000	3197.100000
R4U	17	807.6823529	594.049946	13730.60000	124.60000000	2069.900000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 17

	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
NCSL	1.00000	0.90476	0.96248	0.98766	0.98837	0.98802	0.98676	0.94326	0.93875	0.95202	0.94203
	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
VGC	0.90476	1.00000	0.85855	0.86399	0.86449	0.86320	0.86042	0.79356	0.79330	0.80991	0.78987
	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0002
HE	0.96248	0.85855	1.00000	0.98578	0.98447	0.98486	0.98610	0.86324	0.85408	0.88199	0.86195
	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R1	0.98766	0.86399	0.98578	1.00000	0.99994	0.99992	0.99997	0.92882	0.92193	0.94079	0.92843
	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R2	0.98837	0.86449	0.98447	0.99994	1.00000	0.99988	0.99988	0.93107	0.92422	0.94307	0.93070
	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R3	0.98802	0.86320	0.98486	0.99992	0.99988	1.00000	0.99991	0.93093	0.92398	0.94170	0.93059
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001
R4	0.98676	0.86042	0.98610	0.99997	0.99988	0.99991	1.00000	0.92829	0.92125	0.94012	0.92795
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001
R1U	0.94326	0.79356	0.86324	0.92882	0.93107	0.93093	0.92829	1.00000	0.99952	0.99434	0.99986
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001
R2U	0.93875	0.79330	0.85408	0.92193	0.92422	0.92398	0.92125	0.99952	1.00000	0.99371	0.99928
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001
R3U	0.95202	0.80991	0.88199	0.94079	0.94307	0.94170	0.94012	0.99434	0.99371	1.00000	0.99398
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001
R4U	0.94203	0.78987	0.86195	0.92843	0.93070	0.93059	0.92795	0.99986	0.99928	0.99398	1.00000
	0.0001	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000

VER=1.3

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
NCSL	9	295.333333	188.580885	2658.00000	53.00000000	589.000000
VGC	9	60.000000	47.444705	540.00000	6.00000000	154.000000
HE	9	1361051.522222	1675325.813059	12249463.70000	54508.70000000	4930184.300000
R1	9	14734.955556	10924.046652	132614.60000	2028.10000000	34239.500000
R2	9	16759.811111	12155.064928	150838.30000	2507.40000000	38426.800000
R3	9	15185.177778	11100.644075	136666.60000	2213.90000000	34876.200000
R4	9	13528.655556	10126.041162	121757.90000	1824.80000000	31693.500000
R1U	9	1139.455556	382.566501	10255.10000	544.00000000	1637.700000
R2U	9	1285.133333	410.448176	11566.20000	640.20000000	1804.500000
R3U	9	1649.900000	613.892057	14849.10000	723.20000000	2514.700000
R4U	9	1047.677778	362.333371	9429.10000	473.50000000	1502.700000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 9

	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
NCSL	1.00000	0.95237	0.92935	0.99054	0.99118	0.99068	0.98989	0.83381	0.82949	0.90615	0.82796
	0.0000	0.0001	0.0003	0.0001	0.0001	0.0001	0.0001	0.0052	0.0057	0.0008	0.0059
VGC	0.95237	1.00000	0.84488	0.91992	0.91919	0.91990	0.91796	0.76225	0.75898	0.80205	0.75260
	0.0001	0.0000	0.0041	0.0004	0.0005	0.0004	0.0005	0.0169	0.0177	0.0093	0.0193
HE	0.92935	0.84488	1.00000	0.96538	0.96372	0.96326	0.96612	0.63756	0.63440	0.77772	0.63161
	0.0003	0.0041	0.0000	0.0001	0.0001	0.0001	0.0001	0.0647	0.0665	0.0136	0.0681
R1	0.99054	0.91992	0.96538	1.00000	0.99993	0.99987	0.99998	0.79997	0.79569	0.88933	0.79430
	0.0001	0.0004	0.0001	0.0000	0.0001	0.0001	0.0001	0.0096	0.0103	0.0013	0.0106
R2	0.99118	0.91919	0.96372	0.99993	1.00000	0.99988	0.99992	0.80303	0.79847	0.89161	0.79749
	0.0001	0.0005	0.0001	0.0001	0.0000	0.0001	0.0001	0.0092	0.0099	0.0012	0.0100
R3	0.99068	0.91990	0.96326	0.99987	0.99988	1.00000	0.99989	0.80439	0.79952	0.89054	0.79887
	0.0001	0.0004	0.0001	0.0001	0.0001	0.0000	0.0001	0.0089	0.0097	0.0013	0.0098
R4	0.98989	0.91796	0.96612	0.99998	0.99992	0.99989	1.00000	0.79952	0.79508	0.88916	0.79388
	0.0001	0.0005	0.0001	0.0001	0.0001	0.0001	0.0000	0.0097	0.0104	0.0013	0.0106
R1U	0.83381	0.76225	0.63756	0.79997	0.80303	0.80439	0.79952	1.00000	0.99864	0.96360	0.99934
	0.0052	0.0169	0.0647	0.0096	0.0092	0.0089	0.0097	0.0000	0.0001	0.0001	0.0001
R2U	0.82949	0.75898	0.63440	0.79569	0.79847	0.79952	0.79508	0.99864	1.00000	0.96586	0.99832
	0.0057	0.0177	0.0665	0.0103	0.0099	0.0097	0.0104	0.0001	0.0000	0.0001	0.0001
R3U	0.90615	0.80205	0.77772	0.88933	0.89161	0.89054	0.88916	0.96360	0.96586	1.00000	0.96187
	0.0008	0.0093	0.0136	0.0013	0.0012	0.0013	0.0013	0.0001	0.0001	0.0000	0.0001
R4U	0.82796	0.75260	0.63161	0.79430	0.79749	0.79887	0.79388	0.99934	0.99832	0.96187	1.00000
	0.0059	0.0193	0.0681	0.0106	0.0100	0.0098	0.0106	0.0001	0.0001	0.0001	0.0000

VER=1.4

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
NCSL	250	79.5120000	118.3505864	19878.00000	4.00000000	708.000000
VGC	250	15.2000000	22.7453159	3800.00000	1.00000000	142.000000
HE	249	197354.7722088	601795.9456648	49141338.28000	16.30000000	4899712.900000
R1	250	3253.8944000	6061.2775999	813473.60000	12.80000000	38219.200000
R2	250	3784.6188000	6904.8115416	946154.70000	3.80000000	45255.500000
R3	250	3447.8468000	6332.8293153	861961.70000	24.00000000	42561.800000
R4	250	2968.5908000	5629.8212743	742147.70000	12.80000000	36315.500000
R1U	250	392.7288000	457.4277927	98182.20000	6.80000000	2910.700000
R2U	250	465.3612000	494.5043433	116340.30000	11.60000000	3136.700000
R3U	250	533.4216000	610.0606336	133355.40000	11.60000000	3674.100000
R4U	250	348.4400000	427.1239761	87110.00000	6.80000000	2754.900000

VER=1.4

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / NUMBER OF OBSERVATIONS

	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
NCSL	1.00000	0.95637	0.90482	0.97308	0.97457	0.97195	0.97268	0.91565	0.90707	0.94006	0.91443
	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	250	250	249	250	250	250	250	250	250	250	250
VGC	0.95637	1.00000	0.83119	0.91854	0.92042	0.91448	0.91688	0.88055	0.87525	0.91079	0.87984
	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	250	250	249	250	250	250	250	250	250	250	250
HE	0.90482	0.83119	1.00000	0.95511	0.95533	0.95465	0.95700	0.79576	0.78381	0.83147	0.79739
	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	249	249	249	249	249	249	249	249	249	249	249
R1	0.97308	0.91854	0.95511	1.00000	0.99887	0.99858	0.99923	0.91920	0.90899	0.94170	0.91743
	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	250	250	249	250	250	250	250	250	250	250	250
R2	0.97457	0.92042	0.95533	0.99887	1.00000	0.99955	0.99953	0.91871	0.90829	0.94112	0.91705
	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	250	250	249	250	250	250	250	250	250	250	250
R3	0.97195	0.91448	0.95465	0.99858	0.99955	1.00000	0.99946	0.92137	0.91094	0.94205	0.91968
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001
	250	250	249	250	250	250	250	250	250	250	250
R4	0.97268	0.91688	0.95700	0.99923	0.99953	0.99946	1.00000	0.91743	0.90697	0.93969	0.91572
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001
	250	250	249	250	250	250	250	250	250	250	250
R1U	0.91565	0.88055	0.79576	0.91920	0.91871	0.92137	0.91743	1.00000	0.99244	0.99249	0.99297
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001
	250	250	249	250	250	250	250	250	250	250	250
R2U	0.90707	0.87525	0.78381	0.90899	0.90829	0.91094	0.90697	0.99244	1.00000	0.98653	0.98551
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001
	250	250	249	250	250	250	250	250	250	250	250
R3U	0.94006	0.91079	0.83147	0.94170	0.94112	0.94205	0.93969	0.99249	0.98653	1.00000	0.98686
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001
	250	250	249	250	250	250	250	250	250	250	250
R4U	0.91443	0.87984	0.79739	0.91743	0.91705	0.91968	0.91572	0.99297	0.98551	0.98686	1.00000
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000
	250	250	249	250	250	250	250	250	250	250	250

VER=1.5

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
NCSL	33	206.4848485	226.088219	6814.00000	9.00000000	722.000000
VGC	33	38.5454545	42.858846	1272.00000	1.00000000	145.000000
HE	33	692010.6212121	1132871.764895	22836350.50000	767.30000000	4558670.100000
R1	33	9475.0969697	11635.702657	312678.20000	40.30000000	40926.100000
R2	33	10841.1575758	13299.206233	357758.20000	21.50000000	47487.800000
R3	33	9924.2393939	12235.338632	327499.90000	212.90000000	44703.700000
R4	33	8723.5090909	10801.414869	287875.80000	132.60000000	38205.900000
R1U	33	962.3969697	870.504125	31759.10000	86.00000000	2995.900000
R2U	33	1075.5333333	930.968643	35492.60000	116.10000000	3226.500000
R3U	33	1283.0060606	1147.884345	42339.20000	116.10000000	3778.900000
R4U	33	922.1151515	852.934053	30429.80000	72.70000000	2838.000000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 33

	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
NCSL	1.00000	0.98329	0.93662	0.98402	0.98337	0.98020	0.98354	0.96412	0.96172	0.97457	0.96918
	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
VGC	0.98329	1.00000	0.87628	0.94949	0.94784	0.94149	0.94837	0.94241	0.94034	0.95620	0.95120
	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HE	0.93662	0.87628	1.00000	0.97321	0.97515	0.97730	0.97429	0.89531	0.88982	0.89998	0.89645
	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R1	0.98402	0.94949	0.97321	1.00000	0.99980	0.99950	0.99995	0.96402	0.96092	0.96938	0.95785
	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R2	0.98337	0.94784	0.97515	0.99980	1.00000	0.99961	0.99978	0.96136	0.95811	0.96640	0.95513
	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R3	0.98020	0.94149	0.97730	0.99950	0.99961	1.00000	0.99963	0.96257	0.95938	0.96627	0.95538
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001
R4	0.98354	0.94837	0.97429	0.99995	0.99978	0.99963	1.00000	0.96354	0.96037	0.96860	0.95768
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001
R1U	0.96412	0.94241	0.89531	0.96402	0.96136	0.96257	0.96354	1.00000	0.99973	0.99688	0.98498
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001
R2U	0.96172	0.94034	0.88982	0.96092	0.95811	0.95938	0.96037	0.99973	1.00000	0.99660	0.98408
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001
R3U	0.97457	0.95620	0.89998	0.96938	0.96640	0.96627	0.96860	0.99688	0.99660	1.00000	0.98557
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001
R4U	0.96918	0.95120	0.89645	0.95785	0.95513	0.95538	0.95768	0.98498	0.98408	0.98557	1.00000
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000

VER=1.6

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
NCSL	51	122.4509804	137.4339570	6245.00000	12.00000000	702.000000
VGC	51	21.5294118	24.0560620	1098.00000	1.00000000	129.000000
HE	51	323398.6627451	755963.3386962	16493331.80000	3356.60000000	4558670.100000
R1	51	4950.7784314	7279.8529878	252489.70000	320.70000000	40926.100000
R2	51	5728.6705882	8350.4616907	292162.20000	421.50000000	47487.800000
R3	51	5221.7058824	7798.6466925	266307.00000	421.50000000	44703.700000
R4	51	4512.0098039	6793.6376077	230112.50000	292.20000000	38205.900000
R1U	51	583.5588235	571.7174292	29761.50000	96.30000000	2995.900000
R2U	51	672.4529412	609.7319455	34295.10000	122.20000000	3226.500000
R3U	51	759.8823529	723.1824768	38754.00000	134.60000000	3778.900000
R4U	51	521.6862745	546.1196265	26606.00000	67.30000000	2838.000000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 51

	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
NCSL	1.00000	0.95423	0.92014	0.97428	0.97375	0.97130	0.97423	0.90192	0.90509	0.91366	0.89900
	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
VGC	0.95423	1.00000	0.85288	0.93482	0.93219	0.92652	0.93307	0.89570	0.89817	0.91140	0.89304
	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HE	0.92014	0.85288	1.00000	0.96804	0.97020	0.97131	0.96954	0.86329	0.86170	0.87004	0.86449
	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R1	0.97428	0.93482	0.96804	1.00000	0.99981	0.99962	0.99991	0.94005	0.94145	0.94968	0.93910
	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R2	0.97375	0.93219	0.97020	0.99981	1.00000	0.99974	0.99976	0.93607	0.93750	0.94624	0.93509
	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R3	0.97130	0.92652	0.97131	0.99962	0.99974	1.00000	0.99964	0.93759	0.93889	0.94661	0.93678
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001
R4	0.97423	0.93307	0.96954	0.99991	0.99976	0.99964	1.00000	0.93827	0.93943	0.94734	0.93744
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001
R1U	0.90192	0.89570	0.86329	0.94005	0.93607	0.93759	0.93827	1.00000	0.99916	0.99528	0.99981
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001
R2U	0.90509	0.89817	0.86170	0.94145	0.93750	0.93889	0.93943	0.99916	1.00000	0.99716	0.99841
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001
R3U	0.91366	0.91140	0.87004	0.94968	0.94624	0.94661	0.94734	0.99528	0.99716	1.00000	0.99398
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001
R4U	0.89900	0.89304	0.86449	0.93910	0.93509	0.93678	0.93744	0.99981	0.99841	0.99398	1.00000
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000

VER=1.7

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
NCSL	11	241.5454545	249.630673	2657.000000	49.00000000	727.000000
VGC	11	47.5454545	49.546672	523.000000	7.00000000	147.000000
HE	11	747757.3454545	1084039.043628	8225330.800000	44562.00000000	3020777.700000
R1	11	10710.6727273	11404.103141	117817.400000	1977.30000000	33229.800000
R2	11	12149.6818182	12871.014940	133646.500000	2381.80000000	37692.400000
R3	11	11115.1909091	11825.667469	122267.100000	2082.40000000	34729.400000
R4	11	9849.6181818	10634.295844	108345.800000	1737.20000000	30936.700000
R1U	11	1139.6727273	915.139692	12536.400000	271.80000000	2647.300000
R2U	11	1271.6272727	980.164662	13987.900000	332.50000000	2859.200000
R3U	11	1516.1090909	1220.071619	16677.200000	333.70000000	3422.100000
R4U	11	1051.8545455	871.633649	11570.400000	234.50000000	2502.300000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 11

	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
NCSL	1.00000	0.99845	0.97258	0.98476	0.98668	0.98260	0.98482	0.91364	0.90795	0.93534	0.91521
	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
VGC	0.99845	1.00000	0.96930	0.98340	0.98477	0.98074	0.98333	0.91662	0.91138	0.93977	0.91791
	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HE	0.97258	0.96930	1.00000	0.98093	0.98417	0.97917	0.98176	0.87085	0.86364	0.88945	0.87360
	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0005	0.0006	0.0002	0.0004
R1	0.98476	0.98340	0.98093	1.00000	0.99966	0.99983	0.99996	0.94633	0.94184	0.95742	0.94797
	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R2	0.98668	0.98477	0.98417	0.99966	1.00000	0.99940	0.99962	0.93868	0.93395	0.95091	0.94039
	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R3	0.98260	0.98074	0.97917	0.99983	0.99940	1.00000	0.99974	0.94865	0.94430	0.95859	0.95027
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001
R4	0.98482	0.98333	0.98176	0.99996	0.99962	0.99974	1.00000	0.94552	0.94084	0.95655	0.94723
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001
R1U	0.91364	0.91662	0.87085	0.94633	0.93868	0.94865	0.94552	1.00000	0.99975	0.99676	0.99995
	0.0001	0.0001	0.0005	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001
R2U	0.90795	0.91138	0.86364	0.94184	0.93395	0.94430	0.94084	0.99975	1.00000	0.99614	0.99952
	0.0001	0.0001	0.0006	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001
R3U	0.93534	0.93977	0.88945	0.95742	0.95091	0.95859	0.95655	0.99676	0.99614	1.00000	0.99660
	0.0001	0.0001	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001
R4U	0.91521	0.91791	0.87360	0.94797	0.94039	0.95027	0.94723	0.99995	0.99952	0.99660	1.00000
	0.0001	0.0001	0.0004	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000

VER=1.8

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
NCSL	9	233.3333333	259.631470	2100.000000	14.00000000	729.000000
VGC	9	46.5555556	51.198416	419.000000	2.00000000	147.000000
HE	9	777704.9222222	1264174.276563	6999344.300000	1583.80000000	3499454.800000
R1	9	10728.4333333	13700.004486	96555.900000	240.70000000	38072.700000
R2	9	12243.8444444	15568.383337	110194.600000	318.00000000	43432.000000
R3	9	11151.7111111	14266.035768	100365.400000	293.80000000	39601.200000
R4	9	9793.1000000	12699.686398	88137.900000	207.20000000	35051.800000
R1U	9	1024.9222222	991.105919	9224.300000	102.10000000	2764.200000
R2U	9	1133.7888889	1051.646009	10204.100000	134.60000000	2967.800000
R3U	9	1339.7555556	1321.131786	12057.800000	130.20000000	3773.800000
R4U	9	954.5222222	951.575816	8590.700000	87.40000000	2625.100000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 9

	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
NCSL	1.00000	0.98764	0.96621	0.99404	0.99428	0.99409	0.99394	0.98341	0.98275	0.98947	0.98306
	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
VGC	0.98764	1.00000	0.93957	0.97174	0.97342	0.97181	0.97203	0.96218	0.96165	0.96848	0.96247
	0.0001	0.0000	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
HE	0.96621	0.93957	1.00000	0.98752	0.98772	0.98748	0.98772	0.92353	0.92116	0.94144	0.92255
	0.0001	0.0002	0.0000	0.0001	0.0001	0.0001	0.0001	0.0004	0.0004	0.0002	0.0004
R1	0.99404	0.97174	0.98752	1.00000	0.99990	0.99991	0.99981	0.96930	0.96806	0.97993	0.96853
	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R2	0.99428	0.97342	0.98772	0.99990	1.00000	0.99993	0.99991	0.96863	0.96731	0.97930	0.96793
	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
R3	0.99409	0.97181	0.98748	0.99991	0.99993	1.00000	0.99994	0.96976	0.96841	0.98006	0.96907
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001	0.0001
R4	0.99394	0.97203	0.98772	0.99981	0.99991	0.99994	1.00000	0.96822	0.96684	0.97863	0.96756
	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001	0.0001
R1U	0.98341	0.96218	0.92353	0.96930	0.96863	0.96976	0.96822	1.00000	0.99993	0.99832	0.99993
	0.0001	0.0001	0.0004	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001	0.0001
R2U	0.98275	0.96165	0.92116	0.96806	0.96731	0.96841	0.96684	0.99993	1.00000	0.99810	0.99977
	0.0001	0.0001	0.0004	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001	0.0001
R3U	0.98947	0.96848	0.94144	0.97993	0.97930	0.98006	0.97863	0.99832	0.99810	1.00000	0.99800
	0.0001	0.0001	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000	0.0001
R4U	0.98306	0.96247	0.92255	0.96853	0.96793	0.96907	0.96756	0.99993	0.99977	0.99800	1.00000
	0.0001	0.0001	0.0004	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0000

VER=1.9

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
NCSL	7	371.285714	237.478220	2599.00000	31.00000000	714.000000
VGC	7	68.714286	39.020141	481.00000	3.00000000	123.000000
HE	7	1509537.100000	1628815.918800	10566759.70000	14584.60000000	4681847.000000
R1	7	18813.371429	14003.239802	131693.60000	1018.10000000	42087.000000
R2	7	21402.614286	16191.496158	149818.30000	1250.70000000	48762.300000
R3	7	19827.128571	15073.463078	138789.90000	1229.00000000	45756.600000
R4	7	17429.600000	13093.327818	122007.20000	873.80000000	39263.300000
R1U	7	1786.057143	909.206628	12502.40000	318.70000000	3074.300000
R2U	7	1927.528571	987.269178	13492.70000	384.00000000	3306.700000
R3U	7	2334.314286	1162.986328	16340.20000	378.60000000	3895.600000
R4U	7	1668.957143	871.923154	11682.70000	275.50000000	2912.400000

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO:RHO=0 / N = 7

	NCSL	VGC	HE	R1	R2	R3	R4	R1U	R2U	R3U	R4U
NCSL	1.00000	0.94951	0.93172	0.98500	0.98383	0.97995	0.98442	0.91564	0.90633	0.92597	0.91729
	0.0000	0.0011	0.0023	0.0001	0.0001	0.0001	0.0001	0.0038	0.0049	0.0028	0.0036
VGC	0.94951	1.00000	0.78192	0.89500	0.88915	0.88083	0.89340	0.89186	0.88190	0.91010	0.89147
	0.0011	0.0000	0.0378	0.0065	0.0074	0.0088	0.0067	0.0070	0.0086	0.0044	0.0070
HE	0.93172	0.78192	1.00000	0.97105	0.97617	0.97915	0.97195	0.85443	0.84859	0.84613	0.85827
	0.0023	0.0378	0.0000	0.0003	0.0002	0.0001	0.0002	0.0143	0.0158	0.0164	0.0134
R1	0.98500	0.89500	0.97105	1.00000	0.99951	0.99919	0.99998	0.93020	0.92623	0.93412	0.93237
	0.0001	0.0065	0.0003	0.0000	0.0001	0.0001	0.0001	0.0024	0.0027	0.0021	0.0022
R2	0.98383	0.88915	0.97617	0.99951	1.00000	0.99942	0.99950	0.92131	0.91677	0.92526	0.92354
	0.0001	0.0074	0.0002	0.0001	0.0000	0.0001	0.0001	0.0032	0.0037	0.0028	0.0030
R3	0.97995	0.88083	0.97915	0.99919	0.99942	1.00000	0.99935	0.92618	0.92157	0.92721	0.92860
	0.0001	0.0088	0.0001	0.0001	0.0001	0.0000	0.0001	0.0027	0.0032	0.0026	0.0025
R4	0.98442	0.89340	0.97195	0.99998	0.99950	0.99935	1.00000	0.93015	0.92616	0.93352	0.93238
	0.0001	0.0067	0.0002	0.0001	0.0001	0.0001	0.0000	0.0024	0.0027	0.0021	0.0022
R1U	0.91564	0.89186	0.85443	0.93020	0.92131	0.92618	0.93015	1.00000	0.99831	0.99485	0.99991
	0.0038	0.0070	0.0143	0.0024	0.0032	0.0027	0.0024	0.0000	0.0001	0.0001	0.0001
R2U	0.90633	0.88190	0.84859	0.92623	0.91677	0.92157	0.92616	0.99831	1.00000	0.99429	0.99820
	0.0049	0.0086	0.0158	0.0027	0.0037	0.0032	0.0027	0.0001	0.0000	0.0001	0.0001
R3U	0.92597	0.91010	0.84613	0.93412	0.92526	0.92721	0.93352	0.99485	0.99429	1.00000	0.99394
	0.0028	0.0044	0.0164	0.0021	0.0028	0.0026	0.0021	0.0001	0.0001	0.0000	0.0001
R4U	0.91729	0.89147	0.85827	0.93237	0.92354	0.92860	0.93238	0.99991	0.99820	0.99394	1.00000
	0.0036	0.0070	0.0134	0.0022	0.0030	0.0025	0.0022	0.0001	0.0001	0.0001	0.0000

APPENDIX C

PROGRAM TO COMPUTE LINES OF CODE

Mar 16 19:27 1989 cntlines.txt Page 1

```
/*=========================================================================
cntlines.c

Purpose:     To find LOC or Ss of a C source file.

Definition: LOC or Ss is the Lines of count of a C source file. This is
            the total number of source lines in the file excluding blank
            and commentary lines.

Author:     Kumar, Sep 20, 1988

=========================================================================*/

#include <stdio.h>
#define TRUE  1
#define FALSE 0
#define BLANK ' '

int n_bl_lines;
int n_com_lines;

int loc;          /* Lines of count */

main() {
    int in_fd;         /* Input file descriptor */
    char line[150];     /* Buffer to collect a single line */
    char in_file[80]; /* Input file name */
    char *temp;        /* Temporary pointer */

    loc         = 0;
    n_bl_lines  = 0;
    n_com_lines = 0;

    printf("\n Enter Input file Name : ");
    gets(in_file);

    in_fd = open(in_file,0);

    if ( in_fd < 0 ) {
        printf("\nError in opening Input file <%s>\n",in_file);
        exit(0);
    }

    while( get_line(in_fd, line) ) {

        temp = line;

        while( *temp == BLANK )
            temp++;

        if ( *temp != '\0' ) {

            if ( prec_com_match( temp ) )
                find_end_comment( in_fd, temp);
            else
                loc++;
```

```
            }

            else
                n_bl_lines++;

        }   /* End WHILE GETLINE */

        printf("\n LINE COUNTS\n");
        printf(" ---- ------\n\n");
        printf(" LOC or Ss                  = %d\n",loc);
        printf(" Number of blank lines     = %d\n",n_bl_lines);
        printf(" Number of comment lines = %d\n\n",n_com_lines);

        close(in_fd);

    }

    /*-----------------------------------------------------------------------
    get_line()

    Purpose:  To read the next line from the input file.

    Synopsis:
              status = get_line(fd,buffer)
              int status;     Return status
              int fd;         Input file descriptor
              char *buffer;   Buffer to read a line.

    Return(s:
              TRUE  - If a line is successfully read
              FALSE - If error or EOF is reached.

    -----------------------------------------------------------------------*/

    get_line(fd, buffer)
    int fd;
    char *buffer;
    {

        int i, end_of_line, rd_flag;
        char c;

        i = 0;
        end_of_line = FALSE;

        if ( ( rd_flag = read(fd,&c,1) ) != 1 )
            return(FALSE);

        if ( c == '\n' ) {
            buffer[i] = '\0';
            return(TRUE);
        }

        while( rd_flag ==1 && !end_of_line ) {

            if ( c == '\n' )
```

```
                end_of_line = TRUE;
            else
                buffer[i++] = c;

            if ( !end_of_line )
               rd_flag = read(fd,&c,1);

        }  /* End WHILE */

        buffer[i] = '\0';
        return(TRUE);

    }

    /*-----------------------------------------------------------------------
    prec_com_match()

    Purpose:  To see if the current line has any comment lines beginning
              in it.

    Synopsis:
              status = prec_com_match( line )
              int status;     return status
              char *line;     Current line

    Return(s):
              TRUE  - matched
              FALSE - No match

    -----------------------------------------------------------------------*/

    #define SLASH '/'
    #define STAR '*'

    prec_com_match( line )
    char *line;
    {

        int matched;

        matched = FALSE;

        while ( *line != '\0' && !matched ) {

            if ( *line == SLASH && *(line+1) == STAR )
                matched = TRUE;
            line++;

        }

        return(matched);

    }

    /*----------------------------------------------------------------------
    find_end_comment()
```

```
     Purpose:  To go to the position in the file where the current
               comment ends.

     Synopsis:
               find_end_comment( fd , line)
               int fd;       File descriptor
               char *line;  Current line

     Return(s):
               None

     Cautions:
               It is assumed there is no syntax error in the source file.

     ----------------------------------------------------------------------*/

     find_end_comment(fd,line)
     int fd;
     char *line;
     {

         char c;
         int matched;
         int rd_flag;
         int end_of_line;
         int code_line;        /* Does the line have code also? */

         matched = FALSE;
         code_line = FALSE;

         if ( !open_comment ( line ) )  {
             loc++;
             code_line = TRUE;
         }

         while ( *line != '\0' ) {

             if ( *line == STAR && *(line+1) == SLASH )
                 matched = TRUE;

             line++;

         }

         if ( matched  ) {
             if ( !code_line )
                 n_com_lines++;
             return;
         }

         rd_flag = read(fd,&c,1);

         end_of_line = FALSE;

         while( rd_flag == 1 && !matched ) {
```

```
        if ( c == '\n' ) {
            n_com_lines++;
            end_of_line = TRUE;
        }

        if ( end_of_line ) {

            do {

                rd_flag = read( fd, &c ,1);

            }    while ( rd_flag == 1 && ( c ==' ' || c == '\t' ) );

            if ( rd_flag == 1 && c == '\n' )
                n_bl_lines++;

            end_of_line = FALSE;

        }

        if ( c == STAR ) {
            rd_flag = read( fd, &c ,1);
            if ( rd_flag == 1 && c == SLASH )
                matched = TRUE;
        }

        if ( !matched )
            rd_flag = read(fd,&c,1);

    }   /* End WHILE */

    n_com_lines++;

}

/*----------------------------------------------------------------------

open_comment()

Purpose:  To check if the current line has a begin comments mark
          in the start of the line.

Synopsis:
          status =  open_comment(line)
          int status;   Return status
          char *line;   Cuurent line

Return(s):
          TRUE  - Matched
          FALSE - No match

----------------------------------------------------------------------*/


open_comment(line)
```

```
char *line;
{

    if ( *line == SLASH )
        if ( *(line+1) == STAR )
            return(TRUE);
    return(FALSE);

}

/*======================================================================*/
```

APPENDIX D

PROGRAM TO COMPUTE MCCABE'S METRIC

```
/*----------------------------------------------------------------

mccabe.c

Purpose:  To calculate McCabe's Cyclomatic Complexity Number of
          a C source file. ( v(G) )

Author:   Kumar, Sep 26, 1988

Rule:
          The cyclomatic complexity number is calculated as

          v(G) = DE + 1

          where DE is the number of decision points like if's,
          for's, while's, etc., This also includes relational
          OR and AND operators. v(G) is decremented by one for
          each GOTO statement.
-------------------------------------------------------------------*/


#include "mccabe.h"       /* defines states, classes and
                             state tables */

main() {
    char in_file[80];     /* Input file name */
    char token[200];      /* Token collected */
    int  in_fd;           /* Input file descriptor */
    int  vg;              /* Cyclomatic number */

    printf("\n Enter input file name : ");
    gets(in_file);

    in_fd = open(in_file,0);

    if ( in_fd < 0 ) {
        printf("\nError opening input file <%s>\n",in_file);
        exit(0);
    }

    vg = 0;

    while ( get_token( in_fd, token ) == VALID_TOKEN ) {

        switch( token_type (token) ) {

            case DECISION:  vg++;
                            break;

            case GOTO:      vg--;
                            break;

            default:        break;

        }    /* End SWITCH */

    }    /* End WHILE get_token() */

    vg++;

    printf("\nMcCabe's Cyclomatic Number for <%s> = <%d>\n",
             in_file,vg);
```

```
    close(in_fd);

}

/*------------------------------------------------------------------
get_token()

Purpose:    To collect one basic token based on the rules set
            by the state table.

Synopsis:
            status = get_token(fd,token)
            int status;    Return status
            int fd;        Input file descriptor
            char *token;   Buffer to collect one token.

Author:     Kumar, Sep 26, 1988

Caution:
            It it assumed there is no syntax error in the source file.
------------------------------------------------------------------*/

get_token(fd,token)
int fd;
char *token;
{

    int i;
    int cur_state;
    int next_state;
    int char_class;
    int rd_flag;
    char c;

    i = 0;
    token[i] = '\0';
    next_state = START;

    while ( next_state < ENDWORD ) {

        rd_flag = read(fd,&c,1);

        if ( rd_flag != 1 )
            char_class = EOF;
        else
            char_class = class_table[c];

        cur_state = next_state;
        next_state = state_table[cur_state][char_class];

        switch ( next_state ) {

            case WORD:
                            token[i++] = c;
                            break;

            case START:
            case FIRST_SLASH:
            case BEG_COM:
            case FIRST_ST:
                            break;

            case FIRST_OR:
            case FIRST_AND:
                            token[i++] = c;
```

```
                                break;

                case ENDWORD:
                                token[i] = '\0';
                                break;

                case ENDWORD_UG:
                                token[i] = '\0';
                                lseek(fd, -1L, 1);
                                break;

                case ENDWORD_CC:
                                token[i++] = c;
                                token[i] = '\0';
                                break;

                case ERR:
                                printf("\nError in state table\n");
                                exit(0);
                                break;

                default:
                                break;

            }    /* End SWITCH */

        }    /* End WHILE next_stae < ENDWORD */

        if ( next_state != STOP )
            return( VALID_TOKEN );
        else
            return( TOKEN_OVER );

}

/*------------------------------------------------------------------
token_type()

Purpose:    To classify the input token in to one of three types.

Synopsis:
            type = token_type( token )
            int type;    Return value
            char *token; Input token

Return(s):
            DECISION  - Decision point
            GOTO      - Goto statement
            NONE      - Other tokens.

Author(s):  Kumar, Sep 26, 1988

------------------------------------------------------------------*/

token_type(token)
char *token;
{

    if ( strcmp(token,"case") == 0 )
        return( DECISION );
    if ( strcmp(token,"for") == 0 )
        return( DECISION );
    if ( strcmp(token,"if") == 0 )
        return( DECISION );
    if ( strcmp(token,"while") == 0 )
        return( DECISION );
```

```
    if ( strcmp(token,"?") == 0 )
        return( DECISION );
    if ( strcmp(token,"||") == 0 )
        return( DECISION );
    if ( strcmp(token,"&&") == 0 )
        return( DECISION );

    if ( strcmp(token,"goto") == 0 )
        return( GOTO );

    return(NONE);
}

/*--------------------------------------------------------------------

mccabe.h

This file defines the character classes and the state table for
calculating the Mcabe's Cyclomatic Complexity Number of a C
source file.

Author: Kumar, Sep 26, 1988

--------------------------------------------------------------------*/

/*--------------------- CHARACTER CLASSES --------------*/

#define AL  0 /* Alphanumeric Characters */
#define EOF 1 /* End of file */
#define SL  2 /* Slash character */
#define ST  3 /* Star Character */
#define OR  4 /* Bitwise OR operator */
#define AND 5 /* Bitwise AND operator */
#define WH  6 /* Equivalent white space characters
                 (For this program)
                 like '+','-','\n','[','<',etc., */
#define QN  7 /* Question mark */
#define ILL 8 /* Illegal characters */

/*-------------------- STATES ---------------*/

#define START        0   /* Start state, Begin to collect a token */
#define WORD         1   /* Collecting a token */
#define FIRST_SLASH  2   /* There is a slash; Maybe this is a begin
                            comment mark */
#define BEG_COM      3   /* Yes...This is a begin comment mark */
#define FIRST_ST     4   /* There is a star. Maybe this is the end
                            comment mark */
#define FIRST_OR     5   /* There is an OR operator. See if
                            there is one more OR operator to
                            make it a token */
#define FIRST_AND    6   /* There is an AND operator. See if there is
                            one more AND operator to make it a token */
#define ENDWORD      7   /* Collected a token */
#define ENDWORD_UG   8   /* Collected a token; Unget last char */
#define ENDWORD_CC   9   /* Collected a token; Add the last character
                            read, to the token */
#define STOP         10  /* End of file reached */
#define ERR          11  /* Error in state table */

/*--------------END STATES DEFINITIONS ------*/

#define MAX_STATES  12
#define MAX_CLASSES  9
```

```
/*------------- TOKEN TYPES ---------*/

#define DECISION  0  /* Decision point */
#define GOTO      1  /* Goto statement */
#define NONE      2

/*------------- CHARACTER CLASS TABLE ---------*/
int class_table[] = {

   WH, ILL, ILL,  WH, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL,  WH,  WH,  WH,  WH,  WH,  WH, AND,  WH,
   WH,  WH,  ST,  WH,  WH,  WH,  WH,  SL,  AL,  AL,
   AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,  WH,  WH,
   WH,  WH,  WH,  QN,  WH,  AL,  AL,  AL,  AL,  AL,
   AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,
   AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,
   AL,  WH,  WH,  WH,  WH,  WH,  WH,  AL,  AL,  AL,
   AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,
   AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,  AL,
   AL,  AL,  AL,  WH,  OR,  WH,  WH, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL, ILL,
  ILL, ILL, ILL, ILL, ILL

};


/*---------- STATE TABLE -----*/

int state_table[MAX_STATES][MAX_CLASSES] = {

/* Class:       AL            EOF      SL            ST
  OR          AND            WH     QN           ILL */

/*Token*/

/*START*/       WORD,         STOP,    FIRST_SLASH, START,
  FIRST_OR,    FIRST_AND,     START, ENDWORD_CC, ERR,

/*WORD*/        WORD,         ENDWORD, FIRST_SLASH, ENDWORD,
  ENDWORD_UG, ENDWORD_UG,    ENDWORD,    ENDWORD_UG, ERR,

/*FIRSTSLASH*/ ENDWORD_UG, ENDWORD, ENDWORD_UG,  BEG_COM,
  ENDWORD_UG, ENDWORD_UG,    ENDWORD_UG, ENDWORD_UG, ERR,

/*BEG_COM*/     BEG_COM,      ERR,     BEG_COM,      FIRST_ST,
  BEG_COM,     BEG_COM,BEG_COM,     BEG_COM,     ERR,

/*FIRST_STAR*/ BEG_COM,       ERR,     ENDWORD,      FIRST_ST,
  BEG_COM,     BEG_COM,BEG_COM,     BEG_COM,     ERR,

/*FIRST_OR*/    ENDWORD_UG, ERR,       ENDWORD_UG,   ENDWORD_UG,
  ENDWORD_CC, ENDWORD_UG,    ENDWORD_UG, ENDWORD_UG, ERR,

/*FIRST_AND*/  ENDWORD_UG, ERR,       ENDWORD_UG,   ENDWORD_UG,
```

```
  ENDWORD_UG, ENDWORD_CC,  ENDWORD_UG, ENDWORD_UG, ERR

};

#define VALID_TOKEN  1
#define TOKEN_OVER  -1

/*===================================================================*/
```

APPENDIX E

PROGRAM TO COMPUTE HALSTEAD'S METRICS

```
/*--------------------------------------------------------------------
halstead.c

Purpose:    To collect Halstead metrics of a C source file.

Author(s):  Kumar, Oct 8, 1988

--------------------------------------------------------------------*/

#include <stdio.h>

#include "halstead.def"      /* defines states, character classes,
                                state table, structures to collect
                                tokens */

#include <math.h>            /* defines mathematical functions */

int     n1;                  /* Number of unique operators */
int     n2;                  /* Number of unique operands */

int     N_1;                 /* Total number of operators */
int     N_2;                 /* Total number of operands */

int     n_proc,              /* Number of process tokens */
        n_ip,                /* Number of input tokens */
        n_out,               /* Number of output tokens */
        cur_class,           /* Current class under this
                                classification */
        n_loop,              /* Number of tokens in LOOPs */
        n_cond,              /* Number of tokens in CONDITIONS */
        n_seq,               /* #tokens in SEQ stmts */
        control_class;       /* Class under type of control */

int     n_uniq_ip,
        n_uniq_out,
        n_uniq_proc,
        n_uniq_loop,
        n_uniq_cond,
        n_uniq_seq;

int     decl_flag;           /* Is the current line part of a
                                declaration? */
int     n_decl;              /* Total number of declaration
                                statements */


OPERATOR_LINK first_operator;   /* Head of the operator list */
OPERAND_LINK first_operand;     /* Head of the operand list */

OPERATOR_LINK first_ip,
first_out,
first_proc,
first_loop,
first_cond,
first_seq;

int n_sc;      /* Number of semi-colons */

char *strcpyn();


main () {
```

```
char    token[150];     /* Buffer to collect a token */
int     rt_state;       /* End state after collecting a token */
char    in_file[80];    /* Input file name */
int     in_fd;          /* Input file descriptor */
int     i;
double  est_lgth,           /* Estimated length */
        est_vol;            /* Estimated volume */
double  log_b2 ();
double  res_com;            /* Residual complexity */
double  res_com2;           /* Residual complexity */
double  prog_level;
double  difficulty;
double  lang_level;
double  effort;
double  prog_time;

first_operator = NULL;
first_operand = NULL;

first_ip = NULL;
first_out = NULL;
first_proc = NULL;
first_loop = NULL;
first_cond = NULL;
first_seq = NULL;

n_decl = 0;
decl_flag = FALSE;

printf ("\n Enter Input File Name : ");
gets (in_file);

in_fd = open (in_file, 0);

if (in_fd < 0) {
    printf ("\nError opening <%s>\n", in_file);
    exit (0);
}

for (i = 0; i < MAX_SYS_OP; i++)
    sys_op_repeat[i] = FALSE;

for (i = 0; i < MAX_SP_OP; i++)
    spec_op_repeat[i] = FALSE;

cur_class = PROCESS;
control_class = SEQ;

n_ip = n_uniq_ip = 0;
n_out = n_uniq_out = 0;
n_proc = n_uniq_proc = 0;
n_loop = n_uniq_loop = 0;
n_cond = n_uniq_cond = 0;
n_seq = n_uniq_seq = 0;
n1 = 0;
n2 = 0;
N_1 = 0;
N_2 = 0;

printf ("\nW O R K I N G");

while ((rt_state = get_token (in_fd, token)) != STOP) {

    switch (rt_state) {
```

```
        case END_TOK:
        case END_TOK_UG:
            if (!check_sys_operators (token))
                check_prog_operand (token);
            break;

        case END_OP_TOK:
        case END_OP_TOK_UG:
            check_op_tok (token);
            break;

        case END_SP_TOK:
        case END_SP_TOK_UG:
            check_sp_op (token);
            break;

        default:
            break;

    }                           /* End SWITCH */

}                               /* END WHILE */

/* N (EST)  = n1 * log n1 + n2 * log n2 V (EST)  = N log n
  where N  = N_1+N_2 and n = n1+n2    */

est_lgth = (double) n1 * log_b2 ((double) n1) +
    (double) n2 * log_b2 ((double) n2);
est_vol = (double) (N_1+N_2)*log_b2((double)(n1+n2));

prog_level = (2.0 / (double)n1)*((double)n2/(double)N_2);
difficulty = 1.0 / prog_level;
lang_level = (prog_level * prog_level) * est_vol;
effort = difficulty * est_vol;
prog_time = (effort / 18.0) / (60.0 * 60.0);

printf ("\rHALSTEAD METRICS");
printf ("\n-------- -------\n\n");
printf ("Number of Unique operators (n1)  = %d\n", n1);
printf ("Number of Unique operands  (n2)  = %d\n", n2);
printf ("Total Number of operators  (N1)  = %d\n", N_1);
printf ("Total Number of operands   (N2)  = %d\n", N_2);
printf ("Total Length of program          = %d\n\n",(N_1+N_2));
printf ("Estimated Length                 = %.1f\n",est_lgth);
printf ("Estimated Volume                 = %.1f\n",est_vol);

/* Calculate Residual complexities */
calc_res_com1 (&res_com);

printf ("\nTable for Residual Complexity\n");
printf ("----- --- --------- ----------\n");
printf ("Counting Strategy:     All Tokens     Unique\n\n");
printf ("R1: %.1f          %.1f\n",
        res_com, est_lgth);

calc_res_com2 (&res_com, &res_com2);

printf ("R2: %.1f          %.1f\n",
        res_com, res_com2);

calc_res_com3 (&res_com, &res_com2);

printf("R3: %.1f           %.1f\n",
        res_com,res_com2);
```

```
    calc_res_com4 (&res_com, &res_com2);

    printf ("R4: %.1f          %.1f\n",
            res_com, res_com2);

    printf ("\n declarative statements : %d\n\n", n_decl);

    printf ("Hit ENTER to continue");
    getchar ();

    printf ("\nProgram level                  = %.4f\n", prog_level);
    printf ("Program difficulty             = %.4f\n", difficulty);
    printf ("Programming Effort             = %.4f\n", effort);
    printf ("Programming time               = %.4f (Hours)\n",
       prog_time);

    printf ("Language Level                 = %.4f\n\n", lang_level);

    printf ("Number of semi-colons          = %d\n\n",n_sc);

    close (in_fd);

}


/*--------------------------------------------------------------------

check_sys_operators()

Purpose: To check whether an operator is one of the language
         supported operators like 'if', 'while', etc.,

Synopsis:
         status = check_sys_operators( token )
         int status;    Return status
         char *token;   Input token

Return(s):
         TRUE  -  One of the system operators.
         FALSE -  No

Author(s): Kumar, Oct 12, 1988

--------------------------------------------------------------------*/


check_sys_operators (token)
char    *token;
{

    int     done;
    int     i;

    i = 0;
    done = FALSE;

    while (i < MAX_SYS_OP && !done) {

        if (strcmp (token, sys_operators[i]) == 0) {

            done = TRUE;
            N_1++;
            if (sys_op_repeat[i] == FALSE) {
                sys_op_repeat[i] = TRUE;
                n1++;
                make_link(OPT_TYPE,token);
```

```
                }
                else
                    update_link(OPT_TYPE,token);

        }                           /* End IF STRCMP.. */

        i++;

    }                               /* End WHILE */

    return (done);

}

/*-----------------------------------------------------------------
check_prog_operand()

Purpose:     To check whether the token collected is a programmer
             defined operand.

Synopsis:
             check_prog_operand( token );
             char *token;    Input token.

Return(s):   None.

Author(s):   Kumar, Oct 12, 1988

-----------------------------------------------------------------*/

check_prog_operand (token)
char    *token;
{

    int     done;
    OPERAND_LINK cur_operand;
    OPERAND_LINK prev_operand;


    /* Is this the first operand collected..? */

    if (first_operand == NULL) {
        first_operand = (OPERAND_LINK)calloc(1,sizeof(OPERAND));
        if (first_operand == NULL) {
            printf ("Cannot allocate memory\n");
            exit (1);
        }
        first_operand->token = strcpyn (token, MAX_TOK_LEN);
        first_operand -> next = NULL;
        first_operand -> freq = 1;
        n2++;
        N_2++;
        return;
    }

    done = FALSE;
    cur_operand = first_operand;

    while (cur_operand != NULL && !done) {

        if(strcmpn(cur_operand->token,token,MAX_TOK_LEN)==0) {
            done = TRUE;
                 cur_operand->freq++;
        }
        else {
            prev_operand = cur_operand;
```

```
            cur_operand = prev_operand -> next;
        }

    }                               /* End WHILE */

    if (!done) {
        cur_operand = (OPERAND_LINK)calloc(1,sizeof(OPERAND));
        if (cur_operand == NULL) {
            printf ("Error allocating memory\n");
            exit (1);
        }
        prev_operand -> next = cur_operand;
        cur_operand -> token = strcpyn (token, MAX_TOK_LEN);
        cur_operand -> next = NULL;
        cur_operand -> freq = 1;
        n2++;
    }

    N_2++;

}

/*----------------------------------------------------------------
check_op_tok()

Purpose:    To check whether the token collected is  an operator
           token.

Synopsis:
            check_op_token ( token )
            char *token;    Input token.

Return(s):  None.

Author(s):  Kumar, Oct 12, 1988

----------------------------------------------------------------*/

check_op_tok (token)
char    *token;
{

    int     done;
    OPERATOR_LINK cur_operator;
    OPERATOR_LINK prev_operator;


    /* Is this the first operator collected..? */

    if (first_operator == NULL) {
        first_operator = (OPERATOR_LINK)calloc(1,sizeof(OPERATOR));
        if (first_operator == NULL) {
            printf ("Cannot allocate memory\n");
            exit (1);
        }
        first_operator -> token = strcpyn (token, MAX_TOK_LEN);
        first_operator -> next = NULL;
        first_operator -> freq = 1;
        n1++;
        N_1++;
        return;
    }

    done = FALSE;
    cur_operator = first_operator;
```

```
    while (cur_operator != NULL && !done) {

        if (strcmpn(cur_operator->token,token,MAX_TOK_LEN)==0) {
            done = TRUE;
                  cur_operator->freq++;
        }
        else {
            prev_operator = cur_operator;
            cur_operator = prev_operator -> next;
        }

    }                              /* End WHILE */

    if (!done) {
        cur_operator = (OPERATOR_LINK)calloc(1,sizeof(OPERATOR));
        if (cur_operator == NULL) {
            printf ("Error allocating memory\n");
            exit (1);
        }
        prev_operator -> next = cur_operator;
        cur_operator -> token = strcpyn ( token, MAX_TOK_LEN);
        cur_operator -> next = NULL;
        nl++;
        cur_operator -> freq = 1;
    }

    N_1++;

}


/*-----------------------------------------------------------------------
check_sp_op()

Purpose:   Check if the token is one of the special operators
          that stand alone and update Halstead metrics.

Synopsis:  check_sp_op( token )
           char *token;

Return(s):  None

Author(s): Kumar, Oct 12, 1988

---------------------------------------------------------------------*/


check_sp_op (token)
char    *token;
{

    int     done;
    int     i;

    i = 0;
    done = FALSE;

    while (i < MAX_SP_OP && !done) {

        if (strcmp (spec_op[i], token) == 0) {
                 if ( i == 1 )
                        n_sc++;
            done = TRUE;
            N_1++;
            if (spec_op_repeat[i] == FALSE) {
                spec_op_repeat[i] = TRUE;
```

```
                nl++;
                make_link(OPT_TYPE,token);
            }
                 else
                     update_link(OPT_TYPE,token);
        }
        i++;

    }                               /* End WHILE */

}

/*---------------------------------------------------------------
get_token()

Purpose:   To collect one basic token based on the rules set by
          the state table.

Synopsis:
           status = get_token(fd,token)
           int status;    Return status
           int fd;        Input file descriptor
           char *token;   Buffer to collect one token.

Author:    Kumar, Oct 12, 1988

Caution:
           It is assumed there is no syntax error in the
           source file.

---------------------------------------------------------------*/

get_token (fd, token)
int     fd;
char   *token;
{

    int     i;
    int     cur_state;
    int     next_state;
    int     char_class;
    int     rd_flag;
    char    c;

    i = 0;
    token[i] = '\0';
    next_state = START;

    while (next_state < END_TOK) {

        rd_flag = read (fd, &c, 1);

        if (rd_flag != 1)
            char_class = EOF_CL;
        else
            char_class = class_table[c];

        cur_state = next_state;
        next_state = state_table[cur_state][char_class];

        switch (next_state) {

            case FIRST_SL:
            case TOKEN:
            case OP_TOK:
            case SP_TOK:
```

```
case PER:
case DIGIT:
    token[i++] = c;
    break;

case BEG_COM:
case START:
case FIRST_ST:
    if (i > 0) {
        i = 0;
        token[i] = '\0';
    }
    if (c == '\n') {
        cur_class = PROCESS;
        control_class = SEQ;
    }

    if (decl_flag && c == '\n')
        n_decl++;

    break;


case END_TOK:
case END_OP_TOK:
case END_SP_TOK:
    token[i] = '\0';
    update_class (token);
    update_tok (cur_class, control_class, token);

    /* Is there a declaration beginning or
       ending? */
    check_decl (token);

    if (decl_flag && strcmp (token, ";") == 0) {
        n_decl++;
        decl_flag = FALSE;
    }
    if (decl_flag && c == '\n')
        n_decl++;

    break;

case END_TOK_UG:
case END_OP_TOK_UG:
case END_SP_TOK_UG:
    token[i] = '\0';
    update_class (token);
    update_tok (cur_class, control_class, token);

    lseek (fd, -1L, 1);

    /* Is there a declaration beginning or
       ending? */
    check_decl (token);

    if (decl_flag && strcmp (token, ";") == 0) {
        n_decl++;
        decl_flag = FALSE;
    }

    break;

case ERR:
    printf ("\nError in state table\n");
    exit (0);
```

```
                break;

            default:
                break;

        }                           /* End SWITCH */

    }                           /* End WHILE next_state < ENDWORD */

    return (next_state);

}

/*------------------------------------------------------------------
log_b2()

Purpose:    To calculate the logarithm of a number to the base 2.

Synopsis:
            val = log_b2( arg )
            double val;   Return value
            double arg;   Input argument

Author(s):  Kumar, Oct 13, 1988

Caution(s):
            Do not pass a zero value as an argument.

------------------------------------------------------------------*/
double  log_b2 (arg)
double  arg;
{

    return (log10 (arg) / log10 (2.0));

}

/*------------------------------------------------------------------
update_tok()

Purpose:   To update the counts of tokens for the current token
           under the three-way token classification Input, Output
           or Processing tokens.

Synopsis:
           update_tok( class )
           int class; INPUT, OUTPUT  or PROCESS.

Return(s): None.

Author(s): Kumar, Oct 20, 1988

-----------------------------------------------------------------*/


update_tok (class1, class2, token)

int     class1;
int     class2;
char   *token;
{

    switch (class1) {

        case INPUT:
```

```
            n_ip++;
            break;

        case OUTPUT:
            n_out++;
            break;

        case PROCESS:
            n_proc++;
            break;

        default:
            break;

    }

    switch (class2) {

        case LOOP:
            n_loop++;
            break;

        case COND:
            n_cond++;
            break;

        case SEQ:
            n_seq++;
            break;

        default:
            break;

    }

    check_uniq_op (class1, token);
    check_uniq_op (class2, token);

}

/*------------------------------------------------------------------
update_class()

Purpose:    To update the flag cur_class to reflect the state of the
            current token.

Synopsis:
            update_class( word )
            char *word;   Current token.

Return(s): None

Author(s): Kumar, Oct 20, 1988

Note:       cur_class is set to INPUT, OUTPUT or PROCESS.

------------------------------------------------------------------*/

update_class (word)
char   *word;
{

    if (strcmp (word, "read") == 0 ||
            strcmp (word, "scanf") == 0 ||
            strcmp (word, "fscanf") == 0) {
```

```
        cur_class = INPUT;

    }

    else
        if (strcmp (word, "write") == 0 ||
                strcmp (word, "printf") == 0 ||
                strcmp (word, "fprintf") == 0) {

            cur_class = OUTPUT;

        }

    if (strcmp (word, "while") == 0 ||
            strcmp (word, "for") == 0) {

        control_class = LOOP;

    }

    else
        if (strcmp (word, "if") == 0) {

            control_class = COND;

        }

}

/*-----------------------------------------------------------------
calc_res_com1()

Purpose:   To calculate the residual complexity based on the first
            classfication namely OPERATOR-OPERAND.

Synopsis:  calc_res_com1( val1 )
           double *val1; Return complexity value.

Return(s): None

Author(s): Kumar, Oct 20, 1988

Note:      The formula used is
           residual complexity = N1 * log (N1) + N2 * log (N2)

-----------------------------------------------------------------*/
calc_res_com1 (val)
double *val;
{
    *val =
        ((N_1!=0) ? (double)N_1*log_b2((double)N_1):0) +
        ((N_2!=0) ? (double)N_2*log_b2((double)N_2):0);

}
/*-----------------------------------------------------------------
calc_res_com2()

Purpose:   To calculate the residual complexity based on the second
           token classification namely INPUT, OUTPUT and PROCESS
           tokens.(Using both total and unique number of tokens).

Synopsis:
           calc_res_com2( val )
           double *val;   Return complexity value.

Return(s): None
```

```
Author(s): Kumar, Oct 20, 1988

Note:      The formula used here is

           residual complexity = n_ip   * log (n_ip)  +
                                 n_out  * log (n_out) +
                                 n_proc * log (n_proc)

-----------------------------------------------------------------*/
calc_res_com2 (val, val2)
double *val;
double *val2;
{

    *val =
        ((n_ip!=0) ? (double)n_ip*log_b2((double)n_ip):0) +
        ((n_out!=0) ? (double)n_out*log_b2((double)n_out):0) +
        ((n_proc!=0) ? (double)n_proc*log_b2((double)n_proc):0);

    *val2 =
        ((n_uniq_ip!=0) ? (double)n_uniq_ip*
                           log_b2((double)n_uniq_ip):0) +
        ((n_uniq_out!=0) ? (double)n_uniq_out*
                           log_b2((double)n_uniq_out):0) +
        ((n_uniq_proc!=0) ? (double)n_uniq_proc*
                           log_b2((double)n_uniq_proc):0);
}
/*-----------------------------------------------------------------
calc_res_com3()

Purpose:   To calculate the residual complexity based on the third
           token classification namely LOOP, COND and SEQUENTIAL
           tokens. (Using both total and unique number of tokens).

Synopsis:
           calc_res_com3( val )
           double *val;   Return complexity value.

Return(s): None

Author(s): Kumar, Oct 20, 1988

Note:      The formula used here is

           residual complexity = n_loop * log (n_loop) +
                                 n_cond * log (n_cond) +
                                 n_seq  * log (n_seq)

-----------------------------------------------------------------*/
calc_res_com3 (val, val2)
double *val;
double *val2;
{

    *val =
        ((n_loop != 0) ? (double) n_loop *
                   log_b2 ((double) n_loop) : 0) +
        ((n_cond != 0) ? (double) n_cond *
                   log_b2 ((double) n_cond) : 0) +
        ((n_seq != 0) ? (double) n_seq *
                  log_b2 ((double) n_seq) : 0);

    *val2 =
        ((n_uniq_loop != 0) ? (double) n_uniq_loop *
                           log_b2 ((double) n_uniq_loop) : 0) +
```

```
        ((n_uniq_cond != 0) ? (double) n_uniq_cond *
                             log_b2 ((double) n_uniq_cond) : 0) +
        ((n_uniq_seq != 0) ? (double) n_uniq_seq *
                             log_b2 ((double) n_uniq_seq) : 0);

}
/*------------------------------------------------------------------
check_decl()

Purpose:    To check whether the current token begins any
            declarations.

Synopsis:
            check_decl(word)
            char *word;    Input token

Return(s):  None.

Author(s):  Kumar, Oct 25, 1988

Caution(s):
            This gives an approximate indication for the declaration
            statements and does not take care of CAST expressions.
            This limitation is imposed because of the way in which a
            token is collected.

------------------------------------------------------------------*/
check_decl (word)
char    *word;
{

    if (strcmp (word, "char") == 0) {
        decl_flag = TRUE;
        return;
    }
    if (strcmp (word, "int") == 0) {
        decl_flag = TRUE;
        return;
    }
    if (strcmp (word, "short") == 0) {
        decl_flag = TRUE;
        return;
    }
    if (strcmp (word, "long") == 0) {
        decl_flag = TRUE;
        return;
    }
    if (strcmp (word, "unsigned") == 0) {
        decl_flag = TRUE;
        return;
    }
    if (strcmp (word, "double") == 0) {
        decl_flag = TRUE;
        return;
    }
    if (strcmp (word, "float") == 0) {
        decl_flag = TRUE;
        return;
    }
    if (strcmp (word, "struct") == 0) {
        n_decl++;               /* Add one for the brace end */
        decl_flag = TRUE;
        return;
    }
    if (strcmp (word, "union") == 0) {
        n_decl++;               /* Add one for the brace end */
```

```
        decl_flag = TRUE;
        return;
    }
    if (strcmp (word, "typedef") == 0) {
        decl_flag = TRUE;
        return;
    }

}

/*-----------------------------------------------------------------
check_uniq_op()

Purpose:     To check whether the token collected is a unique token.

Synopsis:
             check_uniq_op( class );
             int class;    Type of classification.

Return(s):   None.

Author(s):   Kumar, Oct 12, 1988

-----------------------------------------------------------------*/

check_uniq_op (class, token)
int     class;
char   *token;
{

    int     done;
    int    *uniq_op;
    OPERATOR_LINK cur_operator;
    OPERATOR_LINK prev_operator;
    OPERATOR_LINK * head_ptr;


    switch (class) {

        case INPUT:
            cur_operator = first_ip;
            head_ptr = &first_ip;
            uniq_op = &n_uniq_ip;
            break;

        case OUTPUT:
            cur_operator = first_out;
            head_ptr = &first_out;
            uniq_op = &n_uniq_out;
            break;

        case PROCESS:
            cur_operator = first_proc;
            head_ptr = &first_proc;
            uniq_op = &n_uniq_proc;
            break;

        case LOOP:
            cur_operator = first_loop;
            head_ptr = &first_loop;
            uniq_op = &n_uniq_loop;
            break;

        case COND:
            cur_operator = first_cond;
            head_ptr = &first_cond;
```

```
            uniq_op = &n_uniq_cond;
            break;

        case SEQ:
            cur_operator = first_seq;
            head_ptr = &first_seq;
            uniq_op = &n_uniq_seq;
            break;

        default:
            break;

    }

    /* Is this the first operator collected..? */

    if (cur_operator == NULL) {
        cur_operator = (OPERATOR_LINK)calloc(1,sizeof(OPERATOR));
        if (cur_operator == NULL) {
            printf ("Cannot allocate memory\n");
            exit (1);
        }
        cur_operator -> token = strcpyn ( token, MAX_TOK_LEN);
        cur_operator -> next = NULL;
        *head_ptr = cur_operator;
        return;
    }

    done = FALSE;

    while (cur_operator != NULL && !done) {

        if(strcmpn(cur_operator->token,token,MAX_TOK_LEN)==0) {
            done = TRUE;
        }
        else {
            prev_operator = cur_operator;
            cur_operator = prev_operator -> next;
        }

    }                            /* End WHILE */

    if (!done) {
        cur_operator=(OPERATOR_LINK)calloc(1,sizeof(OPERATOR));
        if (cur_operator == NULL) {
            printf ("Error allocating memory\n");
            exit (1);
        }
        prev_operator -> next = cur_operator;
        cur_operator -> token = strcpyn ( token, MAX_TOK_LEN);
        cur_operator -> next = NULL;
        (*uniq_op)++;
    }


        }
/*-----------------------------------------------------------------
strcpyn( from, to, n)

Purpose:  To copy at most n characters of a string with a null
    termination.

Synopsis: to = strcpyn( from , n)
          char *from;   Source string
                         char *to;     Destination string
```

```
                          int n;        Length to be copied

Return(s): None

Author(s): Kumar, Dec 14, 88

--------------------------------------------------------------------*/

char *strcpyn( from, n)
char *from;
int n;
{
        int i = 0;
        char *to;

        if ( strlen(from) > n-1 )
            *(from+n-1) = '\0';

   if ( (to = malloc(strlen(from)+1) ) == NULL ) {
            printf("\nError X allocating memory\n");
                 exit(1);
        }

        for ( i=0; i<=strlen(from) ; i++)
            *(to+i) = *(from+i);

        return(to);
}
/*------------------------------------------------------------------
strcmpn( strl, str2, n)

Purpose:  To compare utmost n characters of two strings.

Synopsis:
          cmp_flag = strcmpn( strl, str2, n)
                          char *strl;
                          char *str2;
                          int n;
                          int comp_flag;

Return(s):
          0 If two strings are same for n characters
            Relative difference between strl and str2
            in ASCII order.

--------------------------------------------------------------------*/


strcmpn( strl, str2, n)
char *strl;
char *str2;
int n;
{
        if ( strlen(strl) > n-1 )
                *(strl+n-1) = '\0';

        if ( strlen(str2) > n-1 )
                *(str2+n-1) = '\0';

        return ( strcmp(strl,str2) );
}
/*-------------------------------------------------------------
frequency.c

Purpose: To collect the frequncies of the opertors and the
```

```
        operands into their respective structures.

Functions:
          make_link(type,token)
          update_link(type,token)

Details:
        See comments for each funstion

Author(s): Kumar, Nov 16, 1988

----------------------------------------------------------------*/

#include <stdio.h>
#include <ctype.h>
#include "halstead.h"

extern OPERATOR_LINK first_operator;
extern OPERAND_LINK  first_operand;

double log_b2();
char *strcpyn();

FREQ_LINK first_opt_link = NULL,
          first_opd_link = NULL,
          last_opt_link  = NULL,
          last_opd_link  = NULL;  /* Top & Bottom of  lists */

/*-------------------------------------------------------------
make_link()

Purpose:  To create a link for the unique token collected and
    intialize the count to 1.

Synopsis:
          make_link(type, token)
          int type; OPT_TYPE or OPD_TYPE (Operator or Operand)
          char *token;  Token collected

Return(s): None

--------------------------------------------------------------*/

make_link(type, token)
int type;
char *token;
{
    FREQ_LINK cur_link;

    cur_link=(type==OPT_TYPE) ? first_opt_link:first_opd_link;

    if ( cur_link == NULL ) {

        cur_link = (FREQ_LINK) malloc( sizeof( FREQUENCY ) );
        if ( cur_link == NULL ) {
            printf("\nMemory allocation error\n");
            exit(1);
        }
        cur_link->freq = 1;
        cur_link->next = NULL;
        cur_link->token = strcpyn (token, MAX_TOK_LEN);

        if ( type == OPT_TYPE )
            first_opt_link = last_opt_link =cur_link;
        else
            first_opd_link = last_opd_link = cur_link;
```

```
        return;
    }

    cur_link = (FREQ_LINK)malloc( sizeof( FREQUENCY) );
    if ( cur_link == NULL ) {
        printf("Trying to allocate link for <%s>\n",token);
        printf("\nMemory allocation problem\n");
        exit(1);
    }

    cur_link->freq = 1;
    cur_link->next = NULL;
    cur_link->token = strcpyn (token, MAX_TOK_LEN);
    if ( type == OPT_TYPE ) {
        last_opt_link->next = cur_link;
        last_opt_link = cur_link;
    }
    else {
        last_opd_link->next = cur_link;
        last_opd_link = cur_link;
    }

    return;

}

/*-------------------------------------------------------------------

update_link(type,token)

Purpose:  To update the count of a token which is repeated.

Synopsis: update_link(type, token)
          int type; OPT_TYPE or OPD_TYPE (operator or operand)
          char *token;  Token that is repeated

Return(s): None

-------------------------------------------------------------------*/

update_link(type, token)
int type;
char *token;
{
     FREQ_LINK cur_link;
     int done;

     cur_link = (type==OPT_TYPE) ? first_opt_link:first_opd_link;

     done = FALSE;

     while ( cur_link != NULL && !done ) {

         if ( strcmpn(cur_link->token,token,MAX_TOK_LEN) == 0 ) {
             cur_link->freq++;
             done = TRUE;
         }
         cur_link = cur_link->next;

     }

}

debug_prt()
{
   FREQ_LINK cur_link;
```

```
   cur_link = first_opt_link;

   printf("Operators\n");
   while( cur_link != NULL ) {
       printf("%s %d\n",cur_link->token, cur_link->freq);
       cur_link = cur_link -> next;
   }

   printf("Operands\n");
   cur_link = first_opd_link;
   while( cur_link != NULL ) {
       printf("%s %d\n",cur_link->token, cur_link->freq);
       cur_link = cur_link -> next;
   }
}
/*---------------------------------------------------------------------
calc_res_com4()

Purpose: To calculate the residual complexity based on the partition:
         OPERATOR = {Logical, Arithmetic, Others}
         OPERAND  = {Constants, Variables(and others) }

Synopsis: calc_res_com4(val1, val2)
          double *val1; Complexity based on ALL tokens
          double *val2; Complexity based on Unique tokens.

Author(s): Kumar, Nov 18, 1988

---------------------------------------------------------------------*/

calc_res_com4(val1, val2)
double *val1, *val2;
{

    double n_arith = 0.0,
           n_uniq_arith = 0.0,
           n_logical = 0.0,
           n_uniq_logical = 0.0,
           n_others = 0.0,
           n_uniq_others = 0.0,
           n_constants = 0.0,
           n_uniq_constants = 0.0,
           n_vars = 0.0,
           n_uniq_vars = 0.0;


   OPERATOR_LINK cur_opt_link;
        OPERAND_LINK  cur_opd_link;
        FREQ_LINK     cur_link;


   cur_opt_link = first_operator;

   while ( cur_opt_link != NULL ) {

       if ( is_logical(cur_opt_link->token) ) {
           n_logical += cur_opt_link->freq;
           n_uniq_logical++;
       }
       else if ( is_arith(cur_opt_link->token) ) {
                n_arith += cur_opt_link->freq;
                n_uniq_arith++;
            }
            else {
                n_others += cur_opt_link->freq;
```

```
                n_uniq_others++;
            }
        cur_opt_link = cur_opt_link->next;

}
          cur_link = first_opt_link;
while ( cur_link != NULL ) {

        if ( is_logical(cur_link->token) ) {
            n_logical += cur_link->freq;
            n_uniq_logical++;
        }
        else if ( is_arith(cur_link->token) ) {
                n_arith += cur_link->freq;
                n_uniq_arith++;
            }
            else {
                n_others += cur_link->freq;
                n_uniq_others++;
            }
        cur_link = cur_link->next;

}

cur_opd_link = first_operand;
while ( cur_opd_link != NULL ) {

        if ( is_constant(cur_opd_link->token) ) {
            n_constants += cur_opd_link->freq;
            n_uniq_constants++;
        }
        else {
            n_vars += cur_opd_link->freq;
            n_uniq_vars++;
        }
        cur_opd_link = cur_opd_link->next;

}
cur_link = first_opd_link;

while ( cur_link != NULL ) {

        if ( is_constant(cur_link->token) ) {
            n_constants += cur_link->freq;
            n_uniq_constants++;
        }
        else {
            n_vars += cur_link->freq;
            n_uniq_vars++;
        }
        cur_link = cur_link->next;

}

 *vall =
     ((n_logical != 0) ? (double) n_logical *
                          log_b2 ((double) n_logical) : 0) +
     ((n_arith != 0) ? (double) n_arith *
                          log_b2 ((double) n_arith) : 0) +
     ((n_others != 0) ? (double) n_others *
                       log_b2 ((double) n_others) : 0) +
     ((n_constants != 0) ? (double) n_constants *
                          log_b2((double)n_constants):0) +
     ((n_vars != 0) ? (double) n_vars *
                log_b2 ((double) n_vars) : 0);
```

```
    *val2 =
        ((n_uniq_logical != 0) ? (double) n_uniq_logical *
                            log_b2((double)n_uniq_logical):0) +
        ((n_uniq_arith != 0) ? (double) n_uniq_arith *
                            log_b2((double)n_uniq_arith):0) +
        ((n_uniq_others != 0) ? (double) n_uniq_others *
                            log_b2((double)n_uniq_others):0) +
        ((n_uniq_constants != 0) ? (double) n_uniq_constants *
                            log_b2((double)n_uniq_constants):0)+
        ((n_uniq_vars != 0) ? (double) n_uniq_vars *
                          log_b2((double)n_uniq_vars):0);

}

/*-------------------------------------------------------------------
is_logical()

Purpose: To check whether the operator is a logical operator.

Synopsis:
          flag = is_logical(token)
          int flag;
          char *token;

Return(s): TRUE, if logical
           FALSE, otherwise.

Author(s): Kumar, Nov 18,1988

--------------------------------------------------------------------*/

is_logical(token)
char *token;
{


    if ( strcmp( token, "==") == 0 )
        return(TRUE);
    if ( strcmp( token, "<") == 0 )
        return(TRUE);
    if ( strcmp( token, ">") == 0 )
        return(TRUE);
    if ( strcmp( token, "<=") == 0 )
        return(TRUE);
    if ( strcmp( token, ">=") == 0 )
        return(TRUE);
    if ( strcmp( token, "&&") == 0 )
        return(TRUE);
    if ( strcmp( token, "||") == 0 )
        return(TRUE);
    if ( strcmp( token, "!") == 0 )
        return(TRUE);

    return(FALSE);

}

/*------------------------------------------------------------------
is_arith()

Purpose: To check if the operator token is arithmetic.

Synopsis: flag = is_arith(token)
          int flag;
          char *token;
```

```
Return(s): TRUE, if arithmetic
           FALSE, otherwise.

Author(s): Kuamr, Nov 18, 1988

-----------------------------------------------------------------*/

is_arith(token)
char *token;
{

    if ( *token == '<' )
        return(TRUE);
    if ( *token == '>' )
        return(TRUE);
    if ( *token == '=' )
        return(TRUE);
    if ( *token == '+' )
        return(TRUE);
    if ( *token == '-' )
        return(TRUE);
    if ( *token == '*' )
        return(TRUE);
    if ( *token == '/' )
        return(TRUE);
    if ( *token == '¬' )
        return(TRUE);
    if ( *token == '†' )
        return(TRUE);

    return(FALSE);

}

/*-----------------------------------------------------------------
is_constant()

Purpose: To check if the operand token is a constant.

Synopsis: flag = is_constant(token)
          int flag;
          char *token;

return(s): TRUE, if constant
           FALSE, otherwise.

Author(s): Kumar, Nov 18, 1988.

----------------------------------------------------------------*/

is_constant(token)
char *token;
{

    if ( isdigit(*token) )
        return(TRUE);
    else
        return(FALSE);
}

/*--------------------------------------------------------------
halstead.def

This file contains the structure definitions, states,
characters classes and the state table to be used by the
file halstead.c . The state table describes the method to
```

```
collect a basic token and to classify it as an operator or
an operand.

Author(s): Kumar, Oct 8,1988

-------------------------------------------------------------------*/

#include "halstead.h"

char *spec_op[] = {
    "(",
    ";",  /* If any change is made this should still be the second
             element */
    ","
};

    /* Flags to indicate operators repeated */
int spec_op_repeat[MAX_SP_OP];


/*----------------- STATE TABLE --------------------*/

int state_table[MAX_STATES][MAX_CLASS] = {

/* Class:      AL              DG              WH           OP
  SP_OP     SL              ST              PER             ILL
  EOF_CL

/* Token */

/*START*/      TOKEN,          DIGIT,          START,       OP_TOK,
  SP_TOK,   FIRST_SL,       OP_TOK,         PERIOD,         ERR,
  STOP,

/*TOKEN*/      TOKEN,          TOKEN,          END_TOK,     END_TOK_UG,
  END_TOK_UG,    END_TOK_UG,     END_TOK_UG,     END_TOK_UG,      ERR,
  END_TOK,

/*OP_TOK*/     END_OP_TOK_UG, END_OP_TOK_UG, END_OP_TOK, OP_TOK,
  END_OP_TOK_UG, OP_TOK,    OP_TOK,         END_OP_TOK_UG, ERR,
  END_OP_TOK,

/*SP_TOK*/     END_SP_TOK_UG, END_SP_TOK_UG, END_SP_TOK, END_SP_TOK_UG,
  END_SP_TOK_UG, END_SP_TOK_UG, END_SP_TOK_UG, END_SP_TOK_UG, ERR,
  END_SP_TOK,

/*FIRST_SL*/  END_OP_TOK_UG, END_OP_TOK_UG, END_OP_TOK, OP_TOK,
  BEG_COM,  OP_TOK,         BEG_COM,        END_OP_TOK_UG, ERR,
  ERR,

/*BEG_COM*/   BEG_COM,        BEG_COM,        BEG_COM,     BEG_COM,
  BEG_COM,  BEG_COM,        FIRST_ST,       BEG_COM,        ERR,
  ERR,

/*FIRST_ST*/  BEG_COM,        BEG_COM,        BEG_COM,     BEG_COM,
  BEG_COM,  START,          FIRST_ST,        BEG_COM,       ERR,
  ERR,

/*PER*/        END_SP_TOK_UG, DIGIT,          END_SP_TOK, END_SP_TOK_UG,
  END_SP_TOK_UG, END_SP_TOK_UG, END_SP_TOK_UG, END_SP_TOK_UG, ERR,
  ERR,

/*DIGIT*/      END_TOK_UG,     DIGIT,          END_TOK,     END_TOK_UG,
  END_TOK_UG,     END_TOK_UG,     END_TOK_UG,     END_TOK_UG,      ERR,
  ERR
```

```
};

/*---------------------- LAGUAGE SUPPORTED OPERATORS -----------*/

char *sys_operators[MAX_SYS_OP] = {

    "case",
    "char",
    "define",
    "do",
    "double",
    "else",
    "extern",
    "float",
    "for",
    "if",
    "include",
    "int",
    "long",
    "main",
    "short",
    "static",
    "struct",
    "switch",
    "typedef",
    "union",
    "unsigned",
    "while"

};

int sys_op_repeat[MAX_SYS_OP];    /* Are the operators repeated? */

/*------------- CHARACTER CLASS TABLE ---------*/

int class_table[] = {

  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,    WH,
   WH,   ILL,   ILL,    WH,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,    WH,    OP,    OP,    OP,    OP,    OP,    OP,    WH,
SP_OP,    WH,    ST,    OP, SP_OP,    OP,   PER,    SL,    DG,    DG,
   DG,    DG,    DG,    DG,    DG,    DG,    DG,    DG,    OP, SP_OP,
   OP,    OP,    OP,    OP,    WH,    AL,    AL,    AL,    AL,    AL,
   AL,    AL,    AL,    AL,    AL,    AL,    AL,    AL,    AL,    AL,
   AL,    AL,    AL,    AL,    AL,    AL,    AL,    AL,    AL,    AL,
   AL,    OP,    AL,    WH,    OP,    AL,    WH,    AL,    AL,    AL,
   AL,    AL,    AL,    AL,    AL,    AL,    AL,    AL,    AL,    AL,
   AL,    AL,    AL,    AL,    AL,    AL,    AL,    AL,    AL,    AL,
   AL,    AL,    AL, SP_OP,    OP,    WH,    OP,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,   ILL,
  ILL,   ILL,   ILL,   ILL,   ILL
```

```
};

/*-------------------------------------------------------------------
halstead.h

This file contains the structure definitions, states, characters classes
and the state table to be used by the file halstead.c . The state table
describes the method to collect a basic token and to classify it as an
operator or an operand.

Author(s): Kumar, Oct 8,1988

-------------------------------------------------------------------*/

#define TRUE  1
#define FALSE 0

#define MAX_SP_OP     3      /* Maximum number of special operators */
#define MAX_TOK_LEN   20     /* MAximum expected token length */

char *calloc();
char *malloc();

/*--------------------- STRUCTURE DEFINITIONS ------------*/

typedef struct operator {
    char *token;          /* Operator token */
    struct operator *next;
    int freq;
} OPERATOR,    *OPERATOR_LINK;

typedef struct operand {
    char *token;          /* Operand token */
    struct operand *next;
    int freq;
} OPERAND,    *OPERAND_LINK;

/* Structure to collect the operators and operand tokens wtih their
counts */
typedef struct frequency {
    char *token;          /* Operator token or Operand token */
    int freq;
    struct frequency *next;
} FREQUENCY,*FREQ_LINK;

/*--------------------- STATES ---------------*/

#define START           0     /* Start state */
#define TOKEN           1     /* Collecting token */
#define OP_TOK          2     /* Collecting operator token */
#define SP_TOK          3     /* Collecting special token */
#define FIRST_SL        4     /* There is a slash character. Maybe this
                                 is a begin comment mark */
#define BEG_COM         5     /* Yes... This is a begin comment mark */
#define FIRST_ST        6     /* This is the first star. Maybe this is
                                 the end comment mark */
#define PERIOD          7     /* Token starting with '.', Maybe part
                                 of a decimal or not */
#define DIGIT           8     /* Collecting decimal digits */
#define END_TOK         9     /* Collected a token; See if it is an
                                 operator or an operand */
#define END_TOK_UG     10     /* Collected a token; Unget the last
                                 character read */
#define END_OP_TOK     11     /* Collected an operator token */
#define END_OP_TOK_UG  12     /* Collected an operator token; Unget the
                                 last character read */
```

```
#define END_SP_TOK     13     /* Collected a special token */
#define END_SP_TOK_UG  14     /* Collected a special token; Unget the
                                 last character read */
#define STOP           15     /* All operations over */
#define ERR            16     /* Error in state table */

/*----------------- CHARACTER CLASSES -------------*/

#define AL       0           /* Alphabets */
#define DG       1           /* Digits */
#define WH       2           /* White space characters like ' ','\n'
                                '\t', etc., */
#define OP       3           /* Operators like '+','-','&', etc., */
#define SP_OP    4           /* Special operators like ';','(',',' */
#define SL       5           /* Slash character */
#define ST       6           /* Star character '*' */
#define PER      7           /* Period character '.' */
#define ILL      8           /* Illegal character */
#define EOF_CL   9           /* End of file */


#define MAX_STATES  17
#define MAX_CLASS   10

#define MAX_SYS_OP 22

#define INPUT   1
#define OUTPUT  2
#define PROCESS 3

#define LOOP 4
#define COND 5
#define SEQ  6

#define OPT_TYPE 0
#define OPD_TYPE 1

/*==============================================================*/
```

VITA

Chari K. Nandakumar

Candidate for the Degree of

Master of Science

Thesis: QUANTIFYING THE SOFTWARE MAINTENANCE TASK: AN EMPIRICAL STUDY OF COMPLEXITY METRICS ACROSS VERSIONS

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Karaikudi, Tamilnadu, India, May 29, 1964, the son of Mr. and Mrs. R. Krishnamachari. Married to Latha Narasimhan on August 18, 1988.

Education: Graduated from Dr. Alagappa Model High School, Karaikudi, India, in May, 1981; received the Bachelor of Engineering degree with a Major in Electronics and Communication Engineering from the Madurai Kamaraj University, India, in August, 1985; completed requirements for Master of Science degree in Computing and Information Science at Oklahoma State University, in May, 1989.

Professional Experience: Software Engineer, Hindustan Aeronautics Limited, Bangalore, India, 1985-1986; Graduate Assistant, Dept. of Computing and Information Science, Oklahoma State University, Stillwater, Oklahoma, 1987-1989.